CNN-ENABLED VISUAL DATA ANALYTICS AND INTELLIGENT REASONING FOR REAL-TIME OPTIMIZATION AND SIMULATION: AN APPLICATION TO OCCUPANCY-AWARE ELEVATOR DISPATCHING OPTIMIZATION

A Dissertation
Presented to
The Academic Faculty

by

Shu Wang

In Partial Fulfillment
of the Requirements for the Degree
Master in the
Mechanical Engineering

Georgia Institute of Technology
December 2020

CNN-ENABLED VISUAL DATA ANALYTICS AND INTELLIGENT REASONING FOR REAL-TIME OPTIMIZATION AND SIMULATION: AN APPLICATION TO OCCUPANCY-AWARE ELEVATOR DISPATCHING OPTIMIZATION

Approved by:

Dr. Roger Jiao, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Feng Zhou
College of Engineering and Computer Science
University of Michigan-Dearborn

Dr. Seung-Kyum Choi
School of Mechanical Engineering
Georgia Institute of Technology

Dr. A. P. Meliopoulos
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: December 02, 2020

ACKNOWLEDGEMENTS

The work of this thesis would not have been finished without the support of lots of people, and here are the words to thank their contributions.

Firstly, as the academic supervisor of the thesis author, Dr. Roger Jiao provided much constructive advice on the thesis theme. With his help, the idea of capacity estimation during elevator dispatching was extended to situation awareness in real-time operations, and the research can be more generalized as domain modeling: CNN-enabled visual data analytics and reasoning for optimization. He also offered many suggestions on thesis writing to make the thesis more readable and organized.

Secondly, the thesis reading committee members, Dr. Roger Jiao, Dr. Seung-Kyum Choi, Dr. A. P. Meliopoulos, and Dr. Feng Zhou, helped find the limitations of the work and provided constructive suggestions for the revision.

Thirdly, the students studying in the same lab with the author, Xuejian Gong, Pan Zou, Mulang Song, Jianyuan Peng, Yiyun (Cindy) Fei, and Stefan Quaadgras, contributed to the realization of occupancy-aware elevator dispatching. They gave valuable suggestions on the concept design and helped with the CNN applications during the initial development.

Their contributions to this thesis are much appreciated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

CNN	convolutional Neural Network
CBR	case-based reasoning
FOL	first-order logic
LA	linguistic approximation
HC	the passenger handling capacity of the elevator system
IND	indispensable relationship
EGC	elevator group control
$\mathcal{H}$	the desired mapping of a few stacked convolutional layers
$\mathcal{R}$	the residual function of the desired mapping
u	the numerical truth value of a fuzzy set
μ	the grade of membership of a fuzzy set or a rough set
τ	a fuzzy subset of a truth-value set
$\mathcal{F}$	the fuzzy set of a truth-value set
$\mathcal{T}$	a truth-value set
X	object space
x	an object in X
r, w	propositions
I	an information system
U	a non-empty finite set of objects
A	a non-empty finite set of conditional attributes
d	decision attribute
B	a subset of conditional attributes

χ	the set of feasible solutions of a mixed integer programming problem
Pb	the optimization problem
λ	the predicted passenger arrival rate
n_p	the number of passengers in the elevator cab
c	the case feature
ID	the case index
ans	the case solution
w	the weight of a variable
t	the time period
F	the predicted smoothed value
Y	the observation
α	the adaptive smoothing factor
E	the smoothed average error
AE	the smoothed absolute error
e	the error term
M	the mobility value of an object
S	the occupancy area of an object
N	the number of detected objects in an elevator cab
n	the cab sequence of an elevator group control system
C	the estimated capacity of an elevator cab
f	the heuristics state evaluation function in A* search
g	the cost reaching the current state from the initial state in A* search
h	the estimated cost of reach the goal state from the current state in A* search
o	the occupancy factor of an elevator cab environment

d	the cost to answer a hall call as its first hall call task
s	the cost of a trip between a pair of hall calls
p	the hall calls assigned to a cab
P	the number of hall calls assigned to a cab
v_1	the number of hall call tasks before a cab changes the direction
q	the hall call to be assigned
Q	the number of hall calls to be assigned

SUMMARY

For most operational systems, the optimization problem is a combinatorial optimization problem, and the optimization performance largely determines the solution quality. Moreover, there exists a trade-off between the computing time of the decision-making process and the optimization performance, which is particularly evident in a system that conducts real-time operations. To obtain better solutions to the decision-making problem in a shorter time, many optimization algorithms are proposed to improve the searching efficiency in the search space. However, information extraction from the environment is also essential for problem-solving. The environment information not only includes the optimization model inputs, but also contains details of the current situation that may change the problem formulation and optimization algorithm parameter values. Due to the time constraint and the computation time of visual processing algorithms, most conventional operational systems collect environment data from sensor platforms but do not analyze image data, which contains situational information that can assist with the decision-making process. To address this issue, this thesis proposes CNN-enabled visual data analytics and intelligent reasoning for real-time optimization, and a closed-loop optimization structure with discrete event simulation to fit the use of situational information in the optimization model. In the proposed operational system, CNNs are used to extract context information from image data, like the type and the number of objects at the scene. Then reasoning techniques and methodologies are applied to deduct knowledge about the current situation to adjust problem formulation and parameter settings. Discrete event simulation is conducted to test the optimization performance of the system, and adjustments can be made to better fit situational information in the optimization process. To validate the feasibility and effectiveness, an application to occupancy-aware elevator dispatching optimization is presented.

CHAPTER 1. INTRODUCTION

Operations research originally referred to military planners' work during World War II, but now it is the discipline where advanced analytical methods are used to help decision making and obtain the optimal or near-optimal solutions to a problem (Informs, 2020). The applications of operations research involve various fields, including business, industry, society, and so on. Many problem-solving techniques are used in operations research, and mathematical optimization and simulation are essential techniques.

However, because different planning tasks have different time limits, there exists a trade-off between the computing time and the solution performance in real-world applications. Long-term planning operations usually have enough time to collect useful data and obtain the optimal solution, thus focusing more on the solution performance. On the other hand, short-term planning operations are expected to make decisions within the given time, and the time becomes a constraint for finding the optimal solution. For example, workforce scheduling has minutes or hours to obtain the optimal scheduling result, while the elevator dispatching system needs to assign tasks to cabs within seconds. To improve the decision-making efficiency for short-term planning operations, especially those at a real-time scale, optimization and simulation techniques are widely studied, aiming at finding a satisfying solution within a short time.

1.1 Real-time Optimization and Discrete-event Simulation

To obtain the optimal solution to a real-world problem, a mathematical model should be formulated first. The objective function is set based on the performance measuring of a solution. Different optimization algorithms are applied to find the optimal solution. Discrete-event simulation experiments are conducted for simulation-based optimization or for validation of the optimization system.

As the sensing technology and the computing ability becomes more advanced, data can be collected and processed faster, which reduces the computation time of the decision-making process and improves the management efficiency of the operational system. This also satisfies the growing demands of optimization at a real-time scale in various industries to keep pace with the operations. For example, batch manufacturing requires real-time sequencing to fulfill process requirement when tasks need more materials than the storage capacity.

Discrete-event simulation is to model the operations of a system with a sequence of events in time. Because the trigger of an event as well as the event content can be set to follow specified statistical distributions, it can model the uncertainty of the operational system and is often used to validate the optimization model. Discrete-event simulation itself can also be an optimization approach by observing the optimization performance in different parameter settings to find a satisfying solution.

Apart from optimization itself, the time for optimization model input processing, including data measuring, data collection, and data analysis, should also be counted in the decision process. If the optimization model can access more useful information, the solution is expected to have better performance. In other words, a proper procedure for raw data processing can provide high-quality model inputs and is important to making optimal decisions.

1.2 Data Analysis

Data is the information collected through observation (OECD. Publishing, 2008), which can be displayed in either the numeric form or the graphic form. Based on the data form and the goal of the analysis, different methods can be utilized.

Conventional numeric data analysis includes regression analysis and time series analysis. Regression analysis estimates the relationship between a dependent variable and a set of independent variables, while time series analysis focuses on the relationship of the values at different time points of a single variable and the forecast of future data based on previously observed values. Sensors are usually used to obtain numeric data from the environment.

Visual data analysis aims to obtain information from images. In an operational system, sensors may not be the only approach to gain information from the environment, and cameras can capture environment images at a real-time scale. By analysing the contents in an image with convolutional neural networks or machine vision algorithms, more context information can be obtained, including the object type, the object number, and even the object action. By using intelligent reasoning methodologies and techniques, this information can be further utilized to derive situational information.

1.3 Knowledge Representation and Reasoning

Knowledge representation and reasoning are two techniques usually used together, involving the description of knowledge, the acquisition of knowledge from given information, and the application of knowledge to new problems. The definition of knowledge representation is given as the study of available options in the use of a representation scheme to ensure the tractability of reasoning (Levesque, 1986). Its objective is to describe the real-world facts so that machines can draw new conclusions by manipulating the symbolics. Four common knowledge representation techniques are logic, rules, frames, and semantic nets (Davis et al., 1993).

Reasoning can be thought of as the inference process where new expressions or conclusions are obtained from given information and knowledge representations. The assumption that reasoning is valid is that the decision-making process can be understood as mechanical operations over symbolic representations (Levesque, 1986). Different reasoning methodologies are used in various scenarios, and widely used ones include rule-based reasoning, case-based reasoning, fuzzy logic, and rough set theory. Details of knowledge representation and reasoning are introduced in Chapter 5. The objective of using intelligent reasoning in real-time optimization is to gain situational information from the collected data or processed data about the environment.

1.4 Technical Challenges

One limitation of conventional real-time optimization and simulation systems is that they do not make full use of visual data from the environment. This is because traditional visual processing algorithms are efficient in object feature extraction but lack the ability to detect and classify multiple objects in one image quickly. To make the full use of environment images, this thesis proposes to implement CNN-based visual data analytics and intelligent reasoning before optimization, so that the context information and the situational information can be used to assist in the decision-making process. Furthermore, discrete-event simulation experiments are conducted to evaluate the optimization performance. The results can provide feedback to the optimization model, thus adjusting the problem formulation and parameter settings of the optimization model and forming a closed-loop optimization structure.

The technical challenges of this implementation are illustrated as follows. (i) Context information extraction is expected to keep pace with the operations to reflect the changes in the environment in time, thus guaranteeing the timeliness of situational information. (ii) Find the situational information that can influence the solution quality by introducing a factor that represents the current scenario to the optimization model or that can be used to identify different scenarios where new problem formulation can be given. (iii) Modeling the effects of situational information in the optimization model. To correctly represent the function of situational information during decision-making, problem formulation or the objective function parameters should be modified to adapt to the current scenario.

The rest of this thesis is organized as follows. Chapter 2 reviews the related work of CNN-enabled visual data analysis, knowledge representation and reasoning, and combinatorial optimization techniques. Chapter 3 presents the analysis and design of real-time optimization and simulation with CNN-based visual data analysis and reasoning. Chapter 4 discusses CNN-based visual data analysis, with the improved ResNet model for image classification and the Mask R-CNN model for instance segmentation. Chapter 5 introduces reasoning-based methodologies and techniques and related knowledge representation used to extract situational information for optimization. Chapter 6 discusses optimization algorithms in real-time operational systems. Both exact algorithms and heuristic algorithms are introduced, with several algorithms presented in detail: branch-

and-bound, branch-and-cut, branch-and-price, A* search, and genetic algorithms. Chapter 7 introduces the development procedure of discrete-event simulation and analyzes how discrete-event simulation experiments can help enhance the optimization model. In Chapter 8, an application of occupancy-aware elevator dispatching optimization is presented to validate the feasibility of the proposed optimization system. Finally, the contributions of this work and future work are concluded in Chapter 9.

CHAPTER 2. RELATED WORK

2.1 CNN Techniques for Visual Data Analysis

Deep learning-based CNN techniques have been widely studied in recent years. There are many applications of CNN-based visual data analysis, and image classification and object detection are two most common tasks.

Image classification aims to label an image with a particular set of classes. The work of CNN-based image classification starts from the AlexNet in the ILSVRC 2012 competition (Krizhevsky et al., 2017). The AlexNet has five convolutional layers, and it proves the possibility of using convolutional neural networks for multi-class classification. After that, VGGNet and GoogLeNet added more convolutional layers to build a deeper network (Simonyan & Zisserman, 2014; Szegedy et al., 2015), which suggests that deeper network architecture has better learning performance and can improve the prediction accuracy. However, as the network becomes deeper, there are gradient explosion and vanishing problems, influencing the learning performance. These problems are further addressed in the residual neural network (ResNet) by learning the residual between layers (He et al., 2016).

Different from image classification, object detection understands an image as various classes of objects at different positions. Therefore, the outputs of object detection include not only the present objects but their positions. CNN-based object detection starts from Regions with CNN features (R-CNN), which proposes to use selective search to generate region proposals and then to do the classification using CNN features and SVM (Girshick et al., 2014). Since then, improvements are made to both the training and testing process and accuracy. The YOLO removes the original region proposal and uses a convolutional network for both region prediction and class prediction (Redmon et al., 2016). It divides an image into multiple grids and uses grid cells for bounding box prediction. Faster R-CNN uses the region proposal network to replace the selective search and integrates it to form a new network (Ren et al., 2015). Mask R-CNN is based on the framework of Faster R-CNN and adds an RoI Align layer and a branch to predict segmentation masks (He et al., 2017).

2.2 Knowledge Representation and Reasoning Techniques

In the artificial intelligence field, knowledge representation aims at describing the information of the real world using symbolic representations, and reasoning is the formal manipulation of symbols representing a collection of believed propositions to produce new ones (Brachman & Levesque, 2004) or to generate either explicit or implicit conclusions from available knowledge. Case-based reasoning, fuzzy logic, rough set theory are three commonly used reasoning methodologies and techniques.

Case-based reasoning (CBR) originates from the study of using scripts to represent previous situations as knowledge and using plans to understand new situations (Scank & Abelson, 1975). The further work is explored how can the previous situations and situation patterns be used for problem-solving and learning (Watson & Marir, 1994), which is close to the classic definition for CBR: a case-based reasoner solves problems by using or adapting solutions to old problems (Riesbeck & Schank, 2013). Thus, the implementation of CBR does not require much explicit knowledge, and CBR systems can keep learning new knowledge from retained cases (Watson & Marir, 1994).

Fuzzy logic and rough set theory are mathematical approaches to imperfect knowledge and uncertainty (Zbigniew, 2004). Fuzzy logic is based on the truism that much of human reasoning is approximate rather than precise (Zadeh, 1975). It introduces the concept of the fuzzy set, which contains a class of objects with a continuum of grades (Zadeh, 1965). Fuzzy truth-values are used to describe exact description during approximate reasoning, like true, very true, more or less true, by using fuzzy sets. And the fuzzy IF-THEN rules are rules with fuzzy antecedents or fuzzy consequences instead of crisp ones (Dubois & Prade, 1996).

The rough set theory uses set approximations to express vagueness information (Zbigniew, 2004). In this theory, the attributes of all objects follow a finite set, and rough set approximations are defined as the topological operations “interior” and “closure”, corresponding to the lower approximation and the positive region, and the upper approximation and the negative region (Pawlak, 1982). Rough set-based reasoning relies on rules, and rule induction from the dataset is to search relationships among object attributes in the form of production rules (Sabu & Raju, 2011).

2.3 Combinatorial Optimization Algorithms

The combinatorial optimization problem is a process to search for an optimal set of elements from a finite set of items based on an objective function (Schrijver, 2003). Its applications are in many fields: logistics, supply chain optimization, workforce scheduling, vehicle routing, etc. Most combinatorial optimization problems are NP-hard (Hertz & Widmer, 2003). In general, this kind of problem can be solved with heuristic algorithms and exact algorithms.

Exact algorithms are algorithms that can find the optimality of the optimization problem. To solve large scale mixed integer programming, branch-and-bound is broadly used. It is first proposed to address the traveling salesman problem (Little et al., 1963). The branch operation guarantees the integrality of the solution, and the bound operation is used to discard unpromising candidate solutions. There are several variations of branch-and-bound. The cutting-plane method is originally used to refine the feasible space by means of linear inequalities in optimization (Kelley, 1960; Gomory, 1960). The branch-and-cut combines branch-and-bound with the cutting-plane method to tighten the linear programming relaxations (Padberg & Rinaldi, 1991). Another variation is branch-and-price, which combines branch-and-bound and column generation to generate columns that have the potential to improve the objective function when the problem scale is too large (Barnhart & Johnson, 1998), and thus accelerating the searching process.

Heuristic algorithms designate a computational procedure to find a near-optimal solution by iteratively improving a candidate solution based on a given measure of quality (Wang & Chen, 2013). Compared to exact algorithms, heuristic algorithms do not guarantee to find the optimal solution but can find an acceptable or good solution within a reasonable amount of time (Lu & Zhang, 2013). Based on the search strategy, heuristic algorithms can be classified into local search and global search. Local search heuristics tend to be greedier, and they do not totally focus on search but also focus on the movement from one formation to a neighboring refining formation. Typical local search algorithms are A* search (Korf, 1998), tabu search (Glover & Laguna, 1998), variable neighborhood search (Hansen, 1999), and simulated annealing (Kirkpatrick et al., 1983). Global search heuristics are usually population-based heuristics and have efficient methods to escape local optimum. Prevailing algorithms include ant colony optimization (Dorigo et al., 1999), particle swarm optimization (Kennedy et al., 1995), and genetic algorithms (Whitley, 1994).

CHAPTER 3. ANALYSIS AND DESIGN OF REAL-TIME OPTIMIZATION AND SIMULATION WITH CNN-BASED DATA ANALYSIS AND REASONING

3.1 Real-time Visual Data Analytics and Reasoning

Conventional optimization systems usually access environment information with the assistance of a sensor platform, but they lack the visual data analysis of the environment. This is largely because the computing time of visual processing cannot keep up with the optimization at a real-time scale. However, the development in CNN techniques enables fast analysis of image data, including image classification, object detection, pose estimation, and even action recognition, if human beings are an important factor during operations. Such information can be used to comprehend the current situation using reasoning methodologies and techniques. For example, for a fleet management system, if the real-time images of the vehicle environment can be captured and analyzed, information including the road condition, the weather, and the traffic condition (like accidents and traffic jam), can be used to assist in the route planning.

In this regard, this study proposes to conduct visual data analysis and reasoning to extract situational information that can assist the operational system for better optimization performance. The whole process is supported by CNN techniques and artificial intelligence-based reasoning methodologies. The information hierarchy of this process is shown in Figure 3.1.

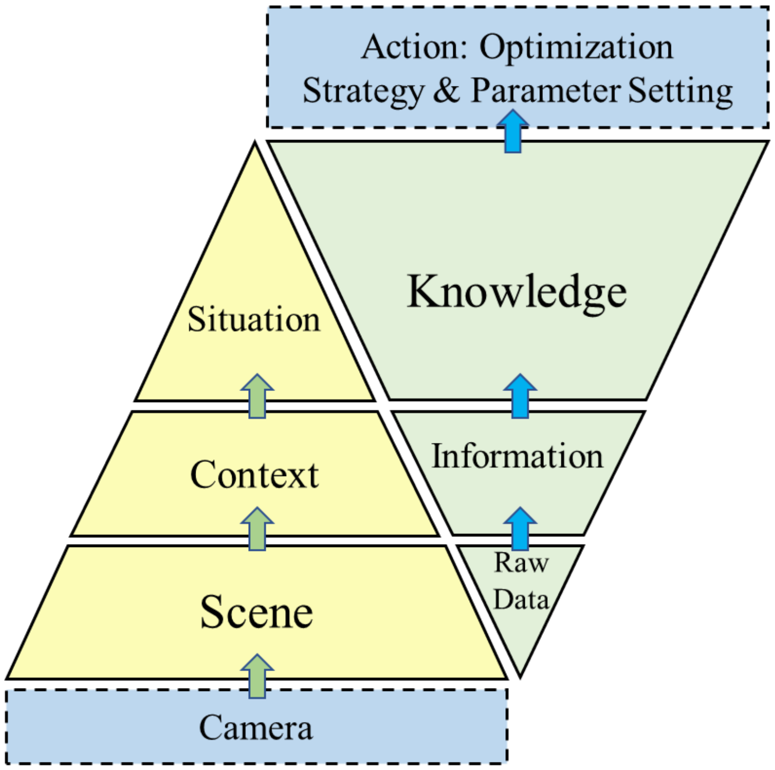


Figure 3.1: Information hierarchy of visual data analysis and reasoning

The information hierarchy of visual data analysis and reasoning is comprised of three levels. The first level is the hardware level, which uses the camera or video monitoring to collect visual data of the environment. This level aims at recording the objects in the scene and is the basis for further analysis. The second level is the context level. Based on how objects in the environment interact with the optimization model, different tasks are implemented: Image classification determines the object type; Object detection finds the number and category of the objects; Action recognition detects the activity that the object is conducting. The third level is the situation level. This level is to refine the obtained information and to understand the underlying story using reasoning methodologies. For instance, in an elevator dispatching system, the context level obtains the real-time number of passengers in the cab. The traffic data is used to recognize the current traffic pattern with fuzzy logic so that an appropriate optimization strategy can be decided to adapt to the changing traffic. Usually, the situational information contains knowledge that can guide the optimization model on optimization variable selection, objective function formulation, algorithm parameter setting, etc.

3.2 Closed-loop Optimization with Simulation Feedback

To validate the effectiveness of the situational information on optimization performance, discrete-event simulation is conducted. In this study, a closed-loop structure is proposed in the validation step to keep adjusting how situational information fits into the optimization model, as shown in Figure 3.2.

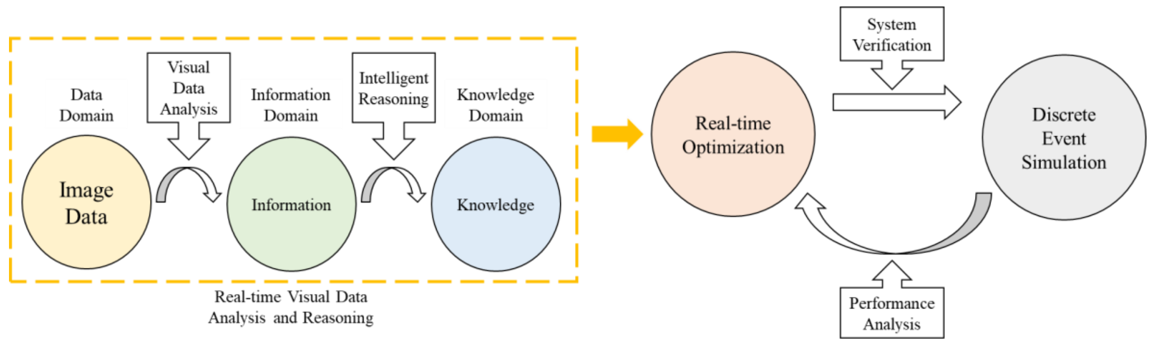


Figure 3.2 Workflow of the proposed optimization system

In a conventional operational system, discrete-event simulation experiments are conducted to verify the system and validate the optimization model. However, when situational information is used during optimization, how it fits into the optimization model should be seriously considered. Comparative experiments between the model with and without the information or the model with different usages can be conducted to test its effectiveness. By analyzing the performance measures and the simulation process, adjustments can be made about the influence of situational information on optimization strategies and variables. Therefore, the optimization model and simulation validation form a closed-loop structure. The performance measures are used as feedback to evaluate current optimization performance and to improve the interactions.

3.3 Optimization System Architecture

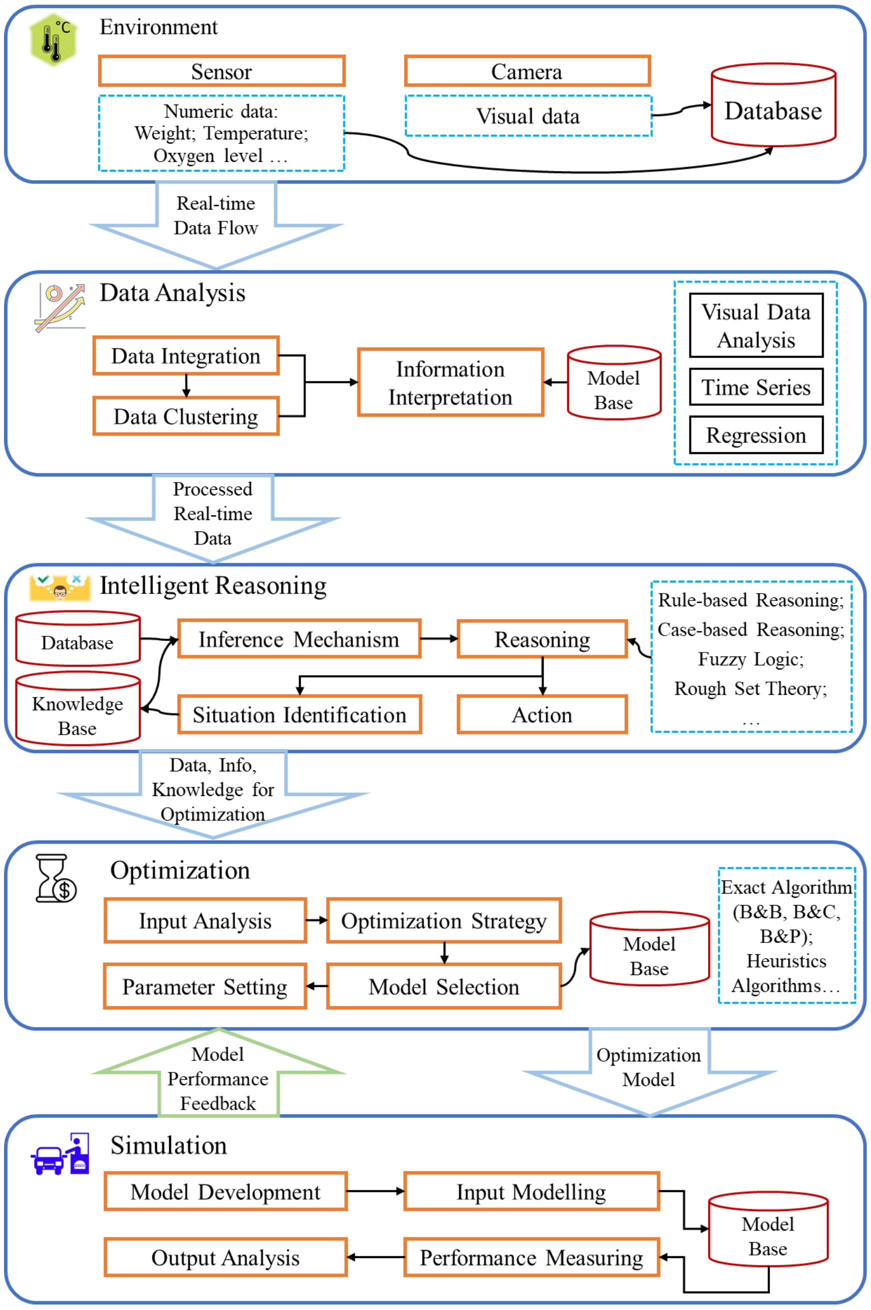


Figure 3.3 Optimization system architecture

The system architecture of the proposed CNN-enabled visual data analytics and reasoning for real-time optimization and simulation is shown in Figure 3.3. The system consists of five layers. The first layer is the environment layer, in which sensors and cameras are used to collect environment data both in numeric and graphic form. The real-time data will then flow into the data analysis layer to further extract context information. The data is firstly integrated and clustered based on its usage and form, and then information interpretation is implemented through several data analysis models: the CNN-based visual data analysis model extracts information about objects in the environment; the time series analysis model predicts future data; the regression analysis model finds some hidden variable values using existing independent variables. Then the processed real-time data is utilized to find the situational information in the intelligent reasoning layer. The inference is firstly conducted with the processed data and existing knowledge for some initial conclusions about the scene. Different reasoning models, including rule-based reasoning, case-based reasoning, fuzzy logic, and so on, are used to deduct useful situational information that can apply existing knowledge for the assistance of optimization. Till this layer, the required data, information, and knowledge for optimization are prepared. In the optimization layer, the input information is integrated: Some data is used to assign values to decision variables; And information and knowledge are used to adjust the optimization strategies, algorithm parameter setting, and optimization problem formulation. An optimization model is selected based on the optimization objective and the time constraint. If the computing time is the priority, the heuristic algorithms can be used. If the task seeks the best solution, exact optimization models can be chosen. The final layer is the simulation layer, with the objective to verify the developed mathematical model for the operational system and to validate the optimization model. Discrete-event simulation experiments are conducted, during which the optimization model is called to provide the optimal solution to the optimization problem. After experiments, the performance measures and experiment results are analyzed, providing feedback to the optimization model and the usage of situational information.

3.4 Chapter Summary

In this chapter, the analysis and design of the proposed optimization system are discussed. The information flow of visual data analytics and reasoning are presented to explain the function of CNNs and reasoning methodologies and how they process the data. The closed-loop optimization structure with discrete-event simulation is proposed, and how simulation interacts and enhances the optimization model are discusses. The optimization system architecture presents the system functional analysis and the operations and required tools or techniques in each stage.

CHAPTER 4. CNN-BASED VISUAL DATA ANALYSIS

4.1 Objectives of CNN-based Data Analysis

In the proposed optimization system, CNNs are used to extract context information from environment images. Different from numeric data analysis, visual data analysis is more object-oriented. The following functionalities can be implemented at a real-time scale based on existing CNN models: image classification, object detection, pose estimation, and action recognition. Therefore, the object type, number, position, posture, and even the activity that the object is conducting in the environment can be obtained. This information is helpful to further understand the current situation with intelligent reasoning.

4.2 Basic CNN Layers

As a hierarchical model, the CNN developed for image classification is usually constructed with three basic layers: convolutional layers, pooling layers, and dense layers.

Convolutional layers are to extract features of images from different classes and create feature maps. In each convolutional layer, there are usually tens or hundreds of convolution kernels to extract different features. Each kernel generates one feature map after the convolution, and the output of the convolutional layer is the collection of feature maps created with the kernels. The convolution operation brings several benefits (Guo et al., 2016). Firstly, the number of parameters is reduced by weight sharing. Secondly, the convolution is invariant to the object location.

The pooling layer is the layer that implements sampling on the feature maps to reduce the feature map dimension and the number of parameters. The pooling process inevitably generates information loss but can speed up the computation. Conventional pooling operations include mean pooling, max pooling, global pooling, stochastic pooling, spatial pyramid pooling, and def-pooling.

The dense layer aims to flatten the 2D matrix into a 1D vector, which is usually used as the output layer for classification. The Softmax function is normally used for multi-class classification, while the Sigmoid function is used for binary classification. The dropout operation is often conducted on the weights of the dense layer during the training to prevent overfitting (Srivastava et al., 2014).

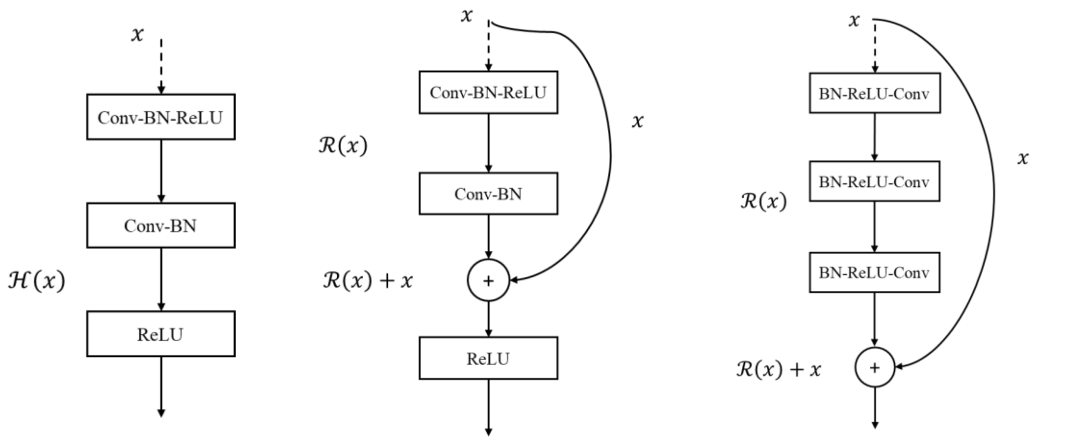
4.3 The Improved ResNet for Image Classification

For conventional deep neural networks, like GoogleNet and VGG16, there are two problems becoming more serious as the network goes deeper: the vanishing gradient problem and the exploding gradient problem. This is because the derivatives will be multiplied when updating the network parameters: If the derivatives are small, the gradient will decrease with the propagation through the model until it vanishes; If the derivatives are large, the gradient will increase with the propagation until it explodes. This makes the training error grows higher when the network is deeper once it reaches some certain levels.

To address this problem, ResNet proposes to use residual learning to every several layers (He et al., 2016), which is implemented with identity mapping by shortcuts, as shown in Figure 4.1(b). Assume the original desired underlying mapping of these stacks in a plain network is $\mathcal{H}(x)$, as shown in Figure 4.1(a). Then the stacked layers are fit the residual mapping $\mathcal{R}(x)$:

$$\mathcal{R}(x) = \mathcal{H}(x) - x \tag{4.1}$$

where the shortcut connection performs the identity mapping. After the recast, the mapping becomes $\mathcal{R}(x) + x$. In this way, the gradient vanishing/exploding problem can be solved.



(a) Plain network (b) Original residual block (c) Improved residual block

Figure 4.1. Plain network and residual blocks

Compared to other models, ResNet is known for its generalization performance and computation cost. The architecture of the ResNet50 is shown in Table 4.1.

Table 4.1 Architecture of ResNet50

Conv1	7*7, 64
Pooling1	3*3, max pooling
Conv2_x	$\begin{bmatrix} 1*1,64 \\ 3*3,64 \\ 1*1,256 \end{bmatrix} * 3$
Conv3_x	$\begin{bmatrix} 1*1,128 \\ 3*3,128 \\ 1*1,512 \end{bmatrix} * 4$

Conv4_x	$\begin{bmatrix} 1*1,256 \\ 3*3,256 \\ 1*1,1024 \end{bmatrix} * 6$
Conv5_x	$\begin{bmatrix} 1*1,512 \\ 3*3,512 \\ 1*1,2048 \end{bmatrix} * 3$
Pooling2	average pooling
Dense	1000 classes

The ResNet model is further improved with the information flow of a residual block (He et al., 2016), which is more proper for image identification. The improved block is shown in Figure 4.1(c). Compared to the original one, batch normalization and the ReLU activation function are moved before 2D convolution. This change further eases the training process and improves generalization.

4.4 Mask R-CNN for Object Detection

Mask R-CNN is a simple and flexible model for instance segmentation, which can generate a segmentation mask for each detected instance. The model is based on the framework of Faster R-CNN, which has a region proposal network (RPN) to generate the region of interest (RoI) with features extracted from the backbone structure. The backbone consists of a bottom-up pathway and a top-bottom pathway. The former is usually a CNN for feature extraction, like ResNet or VGG. The latter is a feature pyramid network that generates semantic features at various resolution scales.

The difference between Faster R-CNN is that Mask R-CNN adds an RoI Align layer and a branch for predicting high-quality segmentation masks. Compared to the Faster R-CNN model in which recognition is after segmentation, the mask branch is in parallel with the branches for classification and bounding box prediction, adding only a small computing cost. Because classification, bounding boxes, and masks are computed simultaneously, the computation time becomes smaller. Furthermore, the RoI Align layer solves the misalignment between the RoI and extracted features introduced by quantization in RoIPool operations, making it possible to predict masks at the pixel scale.

The architecture of Mask R-CNN is presented in Figure 4.2.

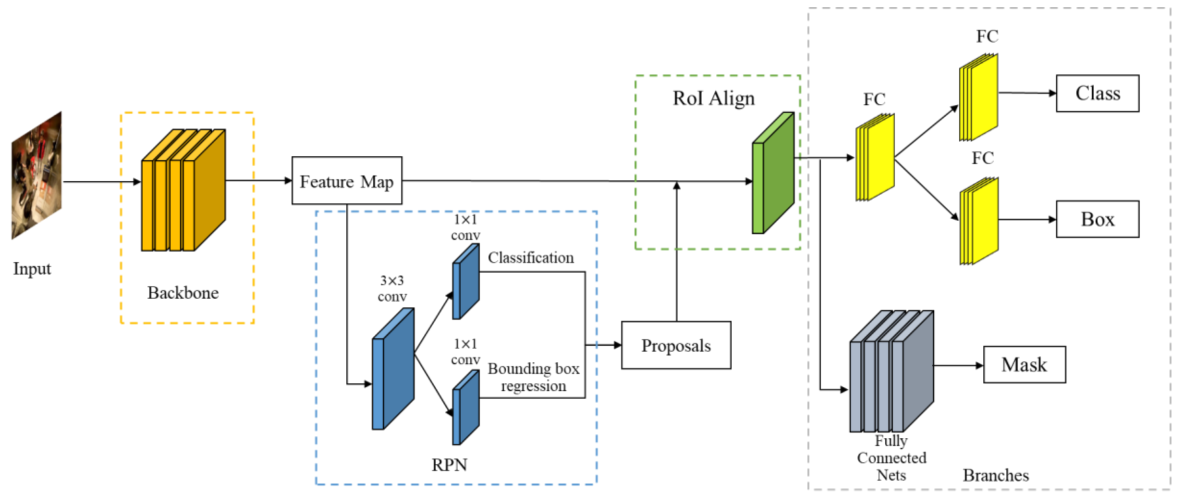


Figure 4.2 Mask R-CNN architecture

4.5 Model Training with Data Augmentation and Transfer Learning

Data augmentation and transfer learning are two training strategies when the dataset is small. The former enlarges the original dataset, while the latter lowers the requirement of the dataset size.

Because of the translation invariance of convolution, geometric transformations can be done on original images, thus generating more data to improve the model generalization performance. Common transformations include flipping, cropping, rotation, translation, and noise injection. Sharpening images with kernel filters is also helpful (Shorten & Khoshgoftaar, 2019). During the model training, combining different augmentation operations is normally taken. However, two issues are worth noticing. The first is that massively inflating the dataset may result in further overfitting when the original dataset is very small. The second is the safety of the augmentation operation. The transformation should not alter the label of the original image. Otherwise, the training will be done with wrongly labeled images.

Transfer learning is to initiate the network parameters with a pre-trained model because the first several convolutional blocks generate the general features that can be used for other classification tasks. Thus, the parameters in these layers can be frozen, and only parameters in the classification layer and the last several layers should be trained. In this way, the training time is largely saved, and less data is needed to develop a new classification model.

4.6 Chapter Summary

This chapter introduces CNN-based visual data analysis techniques. The objective of visual data analysis in the proposed optimization system is discussed. Three basic types of CNN layers are introduced. The principle and architecture of two CNN models, the improved ResNet, and Mask R-CNN, are introduced as state of the art for image classification and object detection tasks. Two common training strategies that lower the requirement of the data sample size and avoid overfitting are also discussed.

CHAPTER 5. REASONING METHODOLOGIES AND TECHNIQUES FOR SITUATIONAL INTELLIGENCE

5.1 Application Scenarios

During optimization input preprocessing, intelligent reasoning is to make inferences about the situational information using existing knowledge and data, thus making the mathematical representation of the real-world scenario more accurate and helping obtain a more satisfying solution for the user. In this chapter, after introducing knowledge representation, three reasoning methodologies are discussed in this chapter: case-based reasoning, fuzzy logic, and rough set theory.

Case-based reasoning can be used when a database that records solutions to past problems can be developed, and the solution to a new problem can be obtained by revising the past solutions. Fuzzy logic is typically used to deal with uncertainty or partial truth. Instead of describing a truth using 0 and 1, fuzzy logic converts the variable into a value between 0 and 1 to describe its degree of truth. Such mathematical description is utilized in approximate reasoning. Rough set theory is often used to handle the uncertainty that the expression of a decision attribute cannot be uniquely defined by other conditional attributes (Pawlak, 1998). Because it approximates a set with a certain attribute using the lower approximation and the upper approximation, a decision table with decision attributes and conditional attributes of objects should be created before rule induction.

5.2 Knowledge Representation

Knowledge representation investigates the expression of knowledge in a computer system (Smedt, 1988). Apart from some data structures like the linked list and the tree structure, four common knowledge representation techniques are logical representations, semantic networks, production rules, and frame representations (JavaTPoint, 2020).

Logical representations use concrete rules to express definite propositions, and the expression can either be syntax or semantics. Syntax refers to the use of legal symbols to express the logic, while semantics focuses on the interpretation of the logic. Logical representations can be categorized into propositional logic and first-order logic (FOL). Propositional logic is a declarative statement consisting of objects, relations, function, and logical connectives. It only conveys a fact is either true or false, and its limitation is that it cannot represent relations like some and all. FOL is an extension of propositional logic and can express complex natural language statements. FOL assumes the world consists of objects, relations, and function, thus representing more complex information than propositional logic. FOL syntax consists of more types of elements, including constant, variable, predicate, function, connective, equality, and quantifier.

Semantic networks use a simple graphical network to represent predicate logic. The network usually consists of nodes and arcs, representing objects and the relationship between objects. The advantage of using semantic networks is that knowledge is easy and transparent to understand, but the disadvantage is that it is not an efficient way for both searching and enlarging the network.

Production rules represent the knowledge in the IF-THEN form and consist of (condition, consequence) pairs. If the conditions for a rule exist, the corresponding consequence or action will be carried out. The advantages of production rules are that they are expressed in natural language and are highly modular, but the execution of rules can be inefficient since the rules are usually in a large number and its management is difficult.

Frame representations are used to describe different entities in the world. The structure consists of a collection of attributes and values, and the attributes can be any type and any size. Frame representations are derived from semantic networks and contain knowledge about an object or an event. Like the semantic network, frame representations are easy to understand and visualize but can bring difficulties to the inference mechanism.

5.3 Case-based Reasoning

Unlike other reasoning methodologies, the advantage of CBR is that it does not require much domain knowledge to solve problems. Instead, it relies on the accumulation of solved cases and finds the solution to its own by comparing the difference of the case features. This makes the CBR system more robust and evolutionary as new cases are continuously put in. Generally, the implementation of CBR involves the following steps: case representation and indexing, new case creation, case retrieval, case reuse or case adaptation, and case retention.

Case representation and indexing is the preparation for further activities. Since CBR relies on knowledge sharing from past cases, a case library should be developed to keep the solved ones. Because each case should represent an experience where knowledge is applied, three types of information can be recorded: the description of the problem, the corresponding solution, and the outcome after the solution is applied. Like a database, each case should be indexed properly for retrieval efficiency. There are several requirements about indexing (Watson, 1994): Indices should be predictive, scalable, and recognizable, and they should suggest the category or the purpose of the case. After the development of the case library, a new case can be encoded when a new situation happens, and the initialization follows the format of case representation.

Then case retrieval is conducted to search the most similar cases in the case library. This process is partially directed by the case indexing and the organization of data. The similarity should be measured with each case in the same category using a specified algorithm. Nearest neighbor is the most common method, which defines different weights for the similarity measuring of different features (Kolodner, 2014).

Case adaptation is to adapt the solution of the retrieved case to the current problem. The adaptation should analyze the prominent differences between cases and apply rules or formulae based on the retrieved solution. There are two types of adaptation: structural adaptation and derivational adaptation. The former is to apply adaptation rules directly to the solution (Kolodner, 2014), while the latter reuses the algorithm or the rule that the retrieved case uses to generate a new solution.

5.4 Fuzzy Logic

Fuzzy logic uses the fuzzy set to describe a situation. A fuzzy set consists of different classes of objects, which can be graded with a membership function to get the value between zero and one (Zadeh, 1965).

Assume X is the space of objects, and x represents a generic object in X. A fuzzy subset of a truth-value set τ in X can be characterized by a membership function μ_τ: $[0,1] \to [0,1]$. $\mu_\tau(x)$ represents in what degree x has the linguistic truth-value τ. For example, let X be the temperature, $\mathcal{T}$ be $\{$cold, warm, hot$\}$, and τ be the fuzzification of $\{$cold$\}$, then $\mu_\tau(x)$ represents the degree of coldness. If τ is not a fuzzy subset, the membership function can be only 0 or 1, with $\mu_\tau(x)$ = 0 or 1 based on whether x belongs to τ or not.

Apart from the membership function that can represent uncertainty, modifiers can be added to the linguistic truth-values in $\mathcal{T}$, like more or less, very, quite, and slightly. These modifiers can affect the membership function. For example, "very" can be defined to square the membership function:

$$\mu_{very\ cold}(x) = \mu_{cold}^2(x) \tag{5.1}$$

Fuzzy logic also follows the standard Łukasiewicz logic (L_1). Let r and w be two propositions, $\neg$ be the negation, $\wedge$ be the conjunction, $\vee$ be the disjunction, $\Rightarrow$ be the implication, the following formulae exist:

$$\mu(\neg r) \triangleq 1 - \mu(\neg r) \tag{5.2}$$

$$\mu(r \wedge w) \triangleq \min\big(\mu(r), \mu(w)\big) \tag{5.3}$$

$$\mu(r \vee w) \triangleq \max\big(\mu(r), \mu(w)\big) \tag{5.4}$$

$$\mu\left(r \Rightarrow w\right) = \min\left(1, 1 - \mu(r) + \mu(w)\right) \quad (5.5)$$

There are two reasons to use the linguistic truth-values of fuzzy logic instead of numerical truth-values of L_1 to do approximate reasoning (Zadeh, 1975). The first is that the truth-value set of fuzzy logic is a countable set, while that of L_1 is a continuum. In most cases, a small finite subset of the truth-values of fuzzy logic is enough for approximate reasoning. The second is that there are more fuzzy propositions than precise propositions in approximate reasoning.

Unlike a conventional proposition, a fuzzy proposition assigns the linguistic value to an object as the value of a variable, instead of using the belong-to relation. For example, a fuzzy proposition r is given by:

$$r \triangleq \text{traffic is } heavy \quad (5.6)$$

Instead of saying traffic is a member of "heavy", it can be interpreted as:

$$Situation(\text{traffic}) = heavy \quad (5.7)$$

in which Situation(traffic) is a variable, and heavy is the assigned linguistic value.

Sometimes, the modified linguistic truth-value τ^* is not in the truth-value set $\mathcal{T}$, and its fuzzy truth-value needs to be approximated by a linguistic truth-value τ in $\mathcal{T}$:

$$\tau^* = LA[\tau] \quad (5.8)$$

in which LA stands for linguistic approximation. For example,

$$u_1 = cold \tag{5.9}$$

$$(u_1, u_2) = approximately\ equal \tag{5.10}$$

$$u_2 = LA[cold \circ approximately\ equal] \tag{5.11}$$

in which $\circ$ denotes the composition of fuzzy relations.

5.5 Rough Set Theory

Rough set-based reasoning relies on the analysis of the table-formed data set, in which each row represents a case, and each column represents an attribute. The table is also referred to as an information system $I = (U, A)$, where U is a non-empty finite set of objects and A is a non-empty finite set of attributes (Zbigniew, 2004). If there is an attribute to describe the class of the object, the information system with the posteriori knowledge is called a decision system, $I = (U, A \cup \{d\})$ where $d \notin A$.

Indiscernibility relation is an important concept in rough set theory, meaning that some objects in the decision table are indiscernible only using a subset of attributes. It can be defined as follows:

$$IND_I(B) = \{(x, x') \in U^2 | \forall a \in B, a(x) = a(x')\} \tag{5.12}$$

where B is a subset of attributes A. It is described as B-indiscernibility relation, and such relation is an equivalence relation. Sets of objects with the same attribute values in this relation can be written as $[x]_B$.

Set approximation is used to describe a set of objects with a certain attribute value. Let $X \subseteq U$. Then a set X can be approximated with B-lower and B-upper approximations, $\underline{B}X$ and $\overline{B}X$:

$$\underline{B}X = \{x | [x]_B \subseteq X\} \tag{5.13}$$

$$\overline{B}X = \{x | [x]_B \cap X = \emptyset\} \tag{5.14}$$

$\underline{B}X$ is called the positive region or lower approximation where objects in $\underline{B}X$ can be classified as members of X without a doubt. $\overline{B}X$ is called the negative region or upper approximation, where objects can possibly be members of X. The boundary region $BN_I(X) = \overline{B}X - \underline{B}X$ contains the objects that cannot be decisively classified based on current knowledge. It is also called the B-boundary region of X. And the region outside the negative region $U - \overline{B}X$ is called the B-outside region of X contains objects that can be classified as not members of X with certainty.

In rough set theory, the membership function $\mu_X^B: U \rightarrow [0,1]$ describes the degree of that x belongs to X in terms of information about x expressed by B-indiscernibility relation:

$$\mu_X^B(x) = \frac{|[x]_B \cap X|}{|[x]_B|} \tag{5.15}$$

In an information system, there might exist redundant attributes. Let $a \in B$, and if $IND_I(B) = IND_I(B - \{a\})$, a is said to be as dispensable. And if all the attributes of B are indispensable, B is said to be independent. A subset $B' \subseteq B$ is a reduct of B if B' is indispensable and $IND_I(B') = IND_I(B)$.

The implementation of RST-based reasoning has several steps: data discretization, attribute reduction, the study of indiscernibility relation, and rule induction. Data discretization is to divide the numeric data into different regions so that numeric data can be converted into nominal data. Conventional data discretization methods include the global discernibility algorithm, which computes the globally semi-optimal cuts using the maximum discernibility heuristic, the quantile-based discretization, and discretization by equal intervals. The attribute reduction is to find the reducts of current conditional attributes from the discernibility matrix. The rejected attributes should be redundant ones whose removal will not worse the classification. The common approach for attribute reduction is the heuristics, while the reduct generation is either based on different criteria, like entropy and discernibility measure, or on a permutation schema over all attributes. The indiscernibility relation refers to a subset of attributes by which two objects are indiscernible. IF-THEN decision rules can then be derived from the indiscernibility classes defined by a subset of attributes.

5.6 Chapter Summary

In this chapter, the function of reasoning methodologies in the optimization system is analyzed. Four knowledge representation techniques are introduced. Three commonly used reasoning methodologies and techniques, case-based reasoning, fuzzy logic, and rough set theory, are discussed in detail, including the application scenario, the reasoning procedure, and the representation of reasoning.

CHAPTER 6. OPTIMIZATION IN REAL-TIME OPERATIONAL SYSTEMS

In a real-time operational system, the optimization model is responsible for finding the optimal decision that meets the requirement or the objectives of the operation. An essential trade-off during this process is between the optimization performance and the execution time. System managers can have different preferences based on the specific application. In this chapter, both exact algorithms and heuristic algorithms are discussed: the former seeks for the optimality in the search space, while the latter sacrifices the optimality for the computation cost. In real-life applications, the solution obtained from exact algorithms and heuristic algorithms may not have much difference in operation costs, but the former may take much longer time. Thus, the problem scale and the performance tolerance between the optimality and the heuristics solution are also two important factors for selecting optimization algorithms.

6.1 Exact Algorithms

Conventional approaches to the exact optimal solution are mainly the branch-and-bound and its variations. The branch-and-bound is a divide-and-conquer method that partitions the problem into independent subproblems, which have smaller feasible regions. The branch-and-cut is the combination of the cutting plane method and the branch-and-bound, in which the former is used to generate valid inequalities in the original problem to narrow down the feasible region. When the problem scale is large, the number of variables can grow exponentially. In this case, it is impossible to formulate the complete model, and column generation will be used to generate variables that can improve the objective function. Branch-and-price is to conduct column generation in the branch-and-bound framework.

6.1.1 Branch-and-Bound

Branch-and-bound takes the divide-and-conquer strategy. Assume the combinatorial optimization problem be in the form (Wolsey, 1998):

$$z = \max\{c^T x | x \in \chi\} \tag{6.1}$$

where χ is the set of feasible solutions. Let $\chi = \chi_1 \cup \ldots \cup \chi_n$ be a decomposition of χ into smaller sets, and

$$\cup_{i=1}^{n} \chi_i = \chi, \cap_{i=1}^{n} \chi_i = \emptyset \tag{6.2}$$

Let $z^k = \max\{c^T x | x \in \chi_k\}$ for $k = 1, \ldots, K$. Then

$$z = \max_{\mathrm{k}} z^k \tag{6.3}$$

Let Pb_{LP} be the linear programming relaxation of the original integer programming problem Pb_{IP}. Decomposition is conducted when the optimal solution to Pb_{LP} contains a non-integral part, where the problem Pb_{LP} can be decomposed as follows:

$$Pb_{LP1} = Pb_{LP} \cap x_j \leq \lfloor \beta_j \rfloor \tag{6.4}$$

$$Pb_{LP2} = Pb_{LP} \cap x_j \geq \lceil \beta_j \rceil \tag{6.5}$$

where x_j is the non-integral variable with the value β_j. Thus, the problem can be solved with a binary tree structure, where each branch is a problem decomposition.

For each subproblem $z^k = \max\{c^T x | x \in \chi_k\}$, there are three scenarios. (i). There is no feasible solution: $\chi_k = \emptyset$; (ii). There is an optimal solution x^i, but the optimal value is no better than the known optimal value: $z^k \leq z$; (iii). There is an optimal solution x^i, but the optimal value should be further computed to know if it is better than the known optimal value. The node representing the subproblem can be discarded when the first or the second scenario exists, which is called pruning.

To study the third scenario, the bound operation is used. The optimal objective function value z has a lower bound $\underline{z}$, which is provided by any feasible integer solution and should be updated whenever a better solution is found. The upper bound is obtained in the linear programming relaxation of the current problem. If an upper bound is smaller than the current best lower bound, then the node can be discarded. The branching and pruning operations are conducted until the feasible solution of the current subproblem equals to $\underline{z}$.

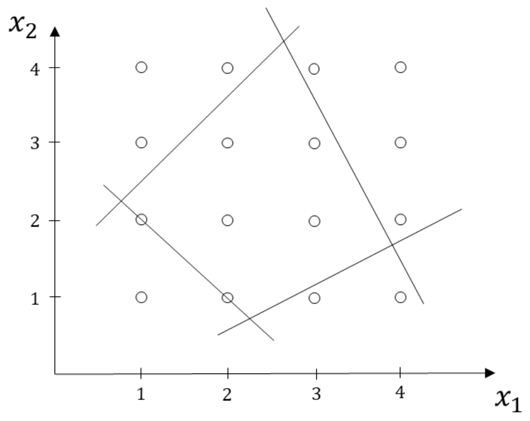
The pseudocode of branch-and-bound is presented in Table 6.1.

Table 6.1 Pseudocode of branch-and-bound

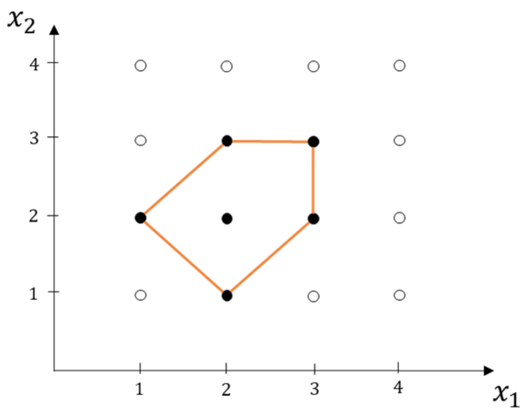
```
// Initialization of the problem set and the current optimality
$S \coloneqq \{Pb_0\}$
$Z \coloneqq -\infty$
while $S \neq \emptyset$
    do
    remove $Pb$ from $S$
    Solve LP($Pb$)
    if LP($Pb$) is feasible
        let $\beta$ be the optimal solution
        if $\beta$ satisfies integrality constraint
            if $c^T\beta > Z$
                store $\beta$
                $Z = \beta$
            else
                // P can be pruned
                if $c^T\beta \leq Z$
                    continue
                end
                let $x_j$ be integer variable with $\beta_j \notin \mathbb{Z}$
                $S \coloneqq S \cup \{P \cap x_j \leq \lfloor \beta_j \rfloor, P \cap x_j \geq \lceil \beta_j \rceil\}$
            end
        end
    end
end
return $Z$
```

6.1.2 Cutting-Plane Method and Branch-and-Cut

For combinatorial optimization problems, the feasible region consists of integer solutions. However, most of the search space consists of fractional solutions, which is the feasible region of the linear programming relaxation of the original problem. To narrow down the search space, the cutting-plane method finds the inequalities in the related problem that excludes the fractional solutions but still contains the original integer solutions, aiming to find the convex hull of the original feasible region until the solution of the relaxed problem is integral. An example of the feasible region of the relaxed problem and its convex hull is shown in Figure 6.1. Figure 6.1(a) presents the feasible region of the relaxed problem, where its integer solutions are included in the polyhedron. Figure 6.1(b) shows the convex hull of the feasible region, where the fractional solutions that exclude the integer solutions are eliminated.



(a) Feasible regions of LP relaxation



(b) Convex hull of the feasible region

Figure 6.1 Polyhedrons formed from the problem feasible region

To achieve this objective, the cutting plane method adds valid inequality whenever the solution of the relaxed problem is not integral to narrow down the search space. Valid inequality is defined as follows:

For a linear programming problem $z = \max\{c^T x | A^T x \leq b, x \in X\}$, where $X = \{x: Ax \leq b, x \in \mathbb{Z}_+^n\}$, $a'^T x \leq b'$ is a valid inequality for $X \subseteq \mathbb{R}^n$ if $a'^T x \leq b', \forall x \in X$.

The adding constraint should exclude the obtained fractional optimal solution but contains all the feasible integer solutions. If the relaxed problem is kept solved with new valid inequalities adding in, all the fractional solutions obtained will be excluded from the feasible region. This operation is stopped until an integral solution is obtained from the relaxed problem. Because the optimal solution to the relaxed problem is also optimal to the problem with integrality constraint, this integer solution is also optimal to the original problem. The pseudocode of the cutting-plane method is shown in Table 6.2.

Table 6.2 Pseudocode of the cutting-plane method

```
// Initialization of the problem set and the current optimality
$t := 0$
$P^0 := P$
Solve $z^t = \max\{c^T x | x \in P^t\}$
Let $x^t$ be the optimal solution
While $x^t \notin \mathbb{Z}^n$
    if $x^t \notin \mathbb{Z}^n$
        Find $(a^t, b^t)$ where $a^t x^t > b^t$ that cuts off $x^t$
        $P^{t+1} = P \cap \{x: a^i x \leq b^i, i = 1, \dots, t\}$
    end
    $t := t + 1$
end
return $x^t$
```

One conventional method to add valid inequalities is through the Gomory's cut (Kelley, 1960).

Because the convergence of the cutting-plane method is slow and keeping adding new cuts makes the relaxed problem very large, it is not commonly used in practice. However, when it is used in the branch-and-bound framework, it can accelerate the search. The combination of the two algorithms is called branch-and-cut. Once a fractional solution is obtained, cuts can be added to the relaxed problem. If an integer solution is found from the new relaxed problem, or its new upper bound is less than the current lower bound, this node will not be branched but discarded sooner.

6.1.3 Column Generation and Branch-and-Price

For large-scale combinatorial optimization problems, it is nearly impossible to enumerate all feasible combinations explicitly. Such problems are often solved by the column generation approach, which is embedded in the branch-and-bound framework. The rationale of column generation is similar to the simplex method. Because non-basic variables of the solution are in the majority, only the variables that have the potential to improve the current solution are generated.

Assume the objective function of the optimization problem is

$$\min\{c^T x | Ax \leq b, x \geq 0\} \tag{6.6}$$

Then its reduced cost can be computed as $c - A^T y$, where y is the dual solution of the linear programming relaxation of the original problem. Finding the variables with minimum reduced costs is then formulated as:

$$\min\{c - A^T y\} \tag{6.7}$$

Usually, this problem can be formulated as a vehicle routing problem with resource constraints.

Therefore, column generation is used to generate new promising variables continuously, while the optimization problem is still solved with branch-and-bound. This combination is called branch-and-price.

Branch-and-price decomposes the original problem into two related problems: the restricted master problem (RMP) that solves the original problem with only a subset of the entire variables, and the pricing problem that finds the variables with minimum negative reduced costs. Figure 6.2 shows the flowchart of branch-and-price.

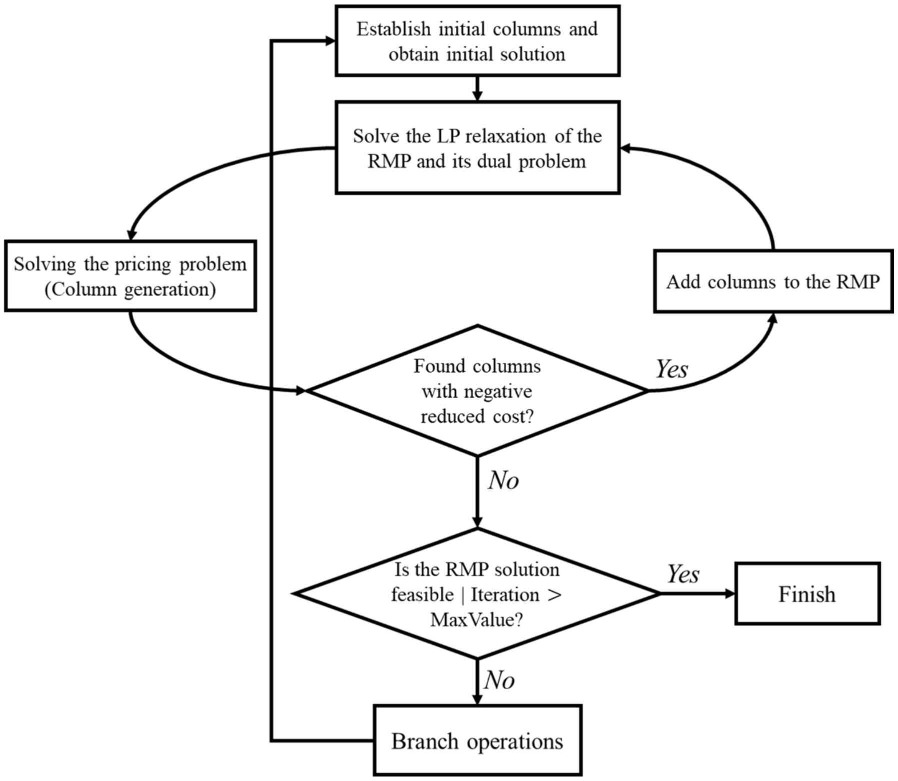

Establish initial columns and obtain initial solution
Solve the LP relaxation of the RMP and its dual problem
Solving the pricing problem (Column generation)
Add columns to the RMP
Found columns with negative reduced cost?
Yes
No
Is the RMP solution feasible | Iteration > MaxValue?
Yes
Finish
No
Branch operations

Figure 6.2 The flowchart of branch-and-price

As discussed above, the pricing problem is usually formulated as a shortest path problem with resource constraints, and the objective is to find the variables with the minimum negative reduced cost. This can be solved by the label correcting algorithm based on dynamic programming. The idea of label correcting is to set the labels for each variable element and track them while extending them to the connectable elements through the graph. After visiting a node, labels on that node will be compared, and the unpromising ones will be discarded with domination rules. The left labels are continuously extended to the destination and form a feasible path. The variables of paths with minimum negative costs are chosen to enter the basis of the RMP.

To guarantee the integrality of the solution, the following branching strategy is taken after column generation is implemented. Assume the optimal solution to the relaxation of the RMP is x_r^*. If $x_r^* \epsilon \mathbb{Z}$, then x_r^* is the optimal solution that is feasible to the original formulation. Otherwise, the fractional part $f_r^* = x_r^* - \lfloor x_r^* \rfloor$ of every variable value is taken to bound the variables with a predefined threshold τ, where $\tau \in (0,1)$. The following strategy is taken:

$$x_r \geq \lceil x_r^* \rceil, \forall r \in R: f_r^* \geq \tau \tag{6.8}$$

However, if $f_r^* < \tau$ for every $r \in R$, then the variable with the largest fractional part will be rounded up:

$$\text{if } r \in R: f_r^* \geq \tau = \emptyset, \text{ then } x_r \geq \lceil x_r^* \rceil, r = argmax_r f_r^* \tag{6.9}$$

6.2 Heuristic Algorithms

In practice, the benefit of finding optimality that has limited improvement than the heuristic solution is not always proportional to the extra computing time. Because of the high computation cost of exact algorithms to solve NP-hard problems, especially when the problem scale is large, there has been a lot of research on heuristic algorithms.

Usually, heuristic algorithms tend to be more greedy than exact algorithms, but they also have the varying capability to escape the local optimum during the search. Thus, they can obtain a near-optimal solution but in shorter time.

Based on the search strategy, heuristics can be classified into local search and global search. Firstly, the local search starts exploring the search space from an initial point, and the initial position will have influences on both the solution quality and the search time, while the global search is less dependent on its initial position. Secondly, global search uses many techniques to search across search space, while local search focuses more on the movement from the current position to its neighboring space.

Conventional local search heuristics include simulated annealing, tabu search, A* search, and so on. And commonly used global search heuristics are particle swarm optimization, genetic algorithms, ant colony optimization, and so on. In this section, A* search is introduced as a local search heuristic, and the genetic algorithm is discussed as a global search heuristic.

6.2.1 A Search: A Local Search Heuristic*

As a local search heuristic, A* search can be understood as an algorithm to find the optimal path, where each path connects an initial state to a goal state (Korf, 1988). During the search, the path is evaluated through a heuristic state evaluation function, f, which is the sum of two cost functions: the sum of the cost reaching the current state from the initial state, g, and the estimated cost of reaching the goal from the current state, h. These two functions are designed based on the specific problem. The search direction is based on the heuristic state evaluation function. The key feature of A* search is that if h is never overestimated, the solution is optimal in terms of g (Korf, 1988).

The procedure of A* search is presented in Table 6.3. In this thesis, A* search is applied to develop the optimization model for elevator dispatching in Chapter 8.5.

Table 6.3 A* Search Algorithm Procedure

```
Initialization
    (1) Set $F$ as the number of tree levels
    (2) Initialize the closed set $C = \emptyset$ and the open node set $M_i = \emptyset$ for
        unsearched nodes in the level $i$, $i = 1, \dots, F$
Search
    (1) Generate the descendant nodes of the node $m_i$, which has the lowest
        heuristic stage function value in the node set $M_i$, and add the descendant
        nodes to $M_{i+1}$
    (2) Calculate the value $f$ for the nodes in $M_{i+1}$
    (3) Sort the nodes in $M_{i+1}$ in increasing order by $f$ values
    (4) Move the node $m_i$ to the closed set $C$
Termination
    (1) If the first node in $M_{i+1}$ is the goal state, end.
    (2) Else, let $i = i + 1$ and return to the search stage.
```

6.2.2 Genetic Algorithm: A Global Search Heuristic

The genetic algorithm (GA) is a population-based iterative optimization algorithm inspired by the process of natural evolution. The solution candidates are treated as a population, and each solution candidate is an individual represented as a chromosome, where each gene represents one element in the solution. During evolution, the objective function is set as the evolution trend. In each iteration, there are several genetic operators to improve the solution quality and to escape the local optimum. The pseudocode of a GA is presented in Table 6.4, and the flowchart is shown in Figure 6.3.

Table 6.4 Pseudocode of a GA

```
// **Initialization of generation 0:**
$ite \coloneqq 0$;
$P_{ite}$ :=initial population of $p$ randomly generated $N$ individuals;
// **Evaluate the fitness value of the populations**
Compute $fit(i)$ for $i \in P_{ite}$;
**while** $ite < maximum\ generation$
    **do**
    // **Create generation** $ite + 1$:
    Select a proportion of members from $P_{ite}$;
    Select another proportion of members to for crossover;
    Combine the selection and offspring;
    Mutate the combined set;
    Compute $fit(i)$ for $i \in P_{ite+1}$;
    $ite \coloneqq ite + 1$;
    **if** the fitness of the best individual is converged
        **return** the best individual
    **end**
**end**
```

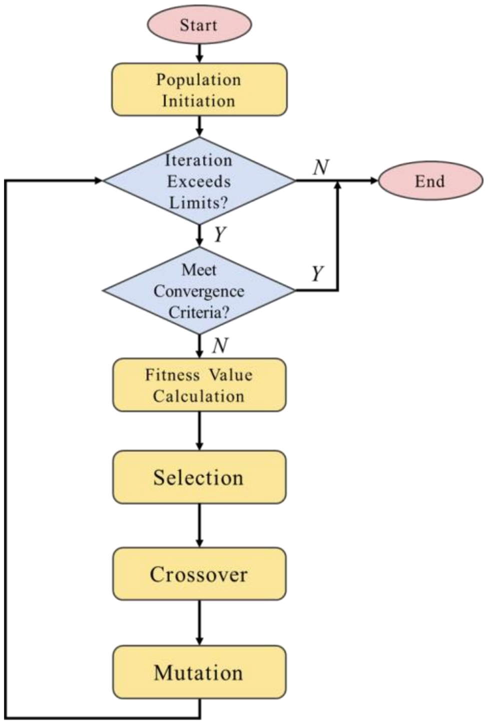
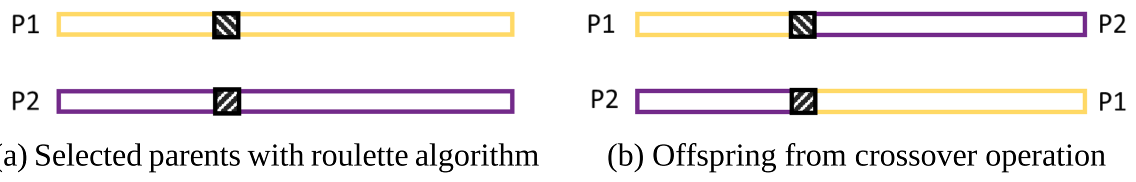

Start
Population Initiation
Iteration Exceeds Limits?
N
End
Y
Meet Convergence Criteria?
Y
N
Fitness Value Calculation
Selection
Crossover
Mutation

Figure 6.3 The flowchart of a GA

The crossover operations aim to exchange part of the parents' chromosomes for producing offspring. Figure 6.4 shows an example of the crossover operation. The operation procedure is as follows. The first step is selecting the parent chromosomes p_1 and p_2 based on their fitness value by applying the roulette algorithm. A fitter individual shows a superior probability of being selected. Then, a random gene index is chosen on the chromosome. The two parents exchange the genes after the chosen index to generate two offspring.



(a) Selected parents with roulette algorithm (b) Offspring from crossover operation

Figure 6.4 The crossover operation

The mutation operations are applied to introduce diversities to the populations. The conventional mutation operation randomly changes the value of a gene at an arbitrary position on the chromosome, as shown in Figure 6.5. The mutation operation will be conducted when a newly generated random number is smaller than the mutation rate, which is a preset constant. This mutation strategy can balance the introduction of diversity to the population and prevent contamination in the late searching stage.

(a) Chromosome before mutation (b) Chromosome after mutation

Figure 6.5 The mutation operation

6.3 Chapter Summary

This chapter introduces both exact algorithms and heuristic algorithms to solve combinatorial optimization problems in operational systems. Comparison is made between exact algorithms and heuristic algorithms. Branch-and-bound, branch-and-cut, and branch-and-price are introduced for exact algorithms, in which the cutting plane method and column generation are two techniques to improve the search efficiency. For heuristic algorithms, the comparison is made between local search heuristics and global search heuristics, and A* search and the genetic algorithms are presented as two typical algorithms.

CHAPTER 7. DISCRETE-EVENT SIMULATION FOR OPTIMIZATION ENHANCEMENTS

Discrete-event simulation can model a complex real-world operational system with a sequence of events in time. In the simulation model, each entity and event are set with different probabilities, trigger conditions, and other properties to represent their existence in the real world. By integrating the optimization model of an operational system into simulation, its effectiveness can be validated.

In this thesis, situational information is introduced to the optimization model through CNN-enabled visual data analytics and reasoning. Because the integration of situational information is flexible, discrete-event simulation can be used to calibrate and improve the optimization model by adjusting the usage of situational information through the simulation feedback, thus forming a closed-loop optimization structure.

7.1 Simulation Model Development

Model development of an operational system can be interpreted as two subproblems, as shown in Figure 7.1: the modeling of operational mechanisms of the real-world problem and the programming of operational algorithms and variable setup. The former ensures the simulation model can operate correctly as the real-world system from the process perspective. The latter is to instantiate the entities and events with specified properties and provide detailed algorithms for the decision-making process.

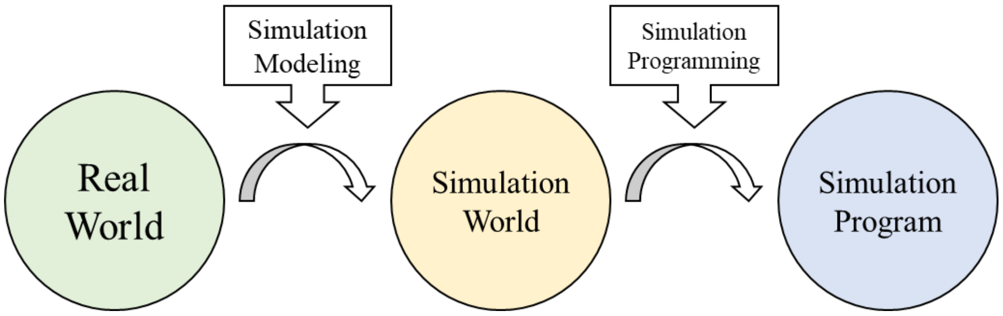


Figure 7.1 A Real-world problem to a simulation model

The workflow of model development is displayed in Figure 7.2. The first step is the problem formulation of the real-world system, including its workflow and mathematical representation. The simulation objective should then be set either to evaluate the performance of the optimization model with certain performance measures or to obtain the optimal solution of a decision-making problem by running experiments with different variable settings. After the preparation, the model entities and corresponding processes are developed based on their operational mechanisms. Real-world data is collected for input modeling. Conventional approaches to process collected data are fitting probability distributions, using data itself, or consulting expert opinions. Optimization algorithms are then embedded in the simulation model. System verification and validation are implemented to test the developed model. The experimental design is implemented to determine the variations of comparative experiments. Finally, experiments are conducted, and results are analyzed to seek the solution to the simulation objectives.

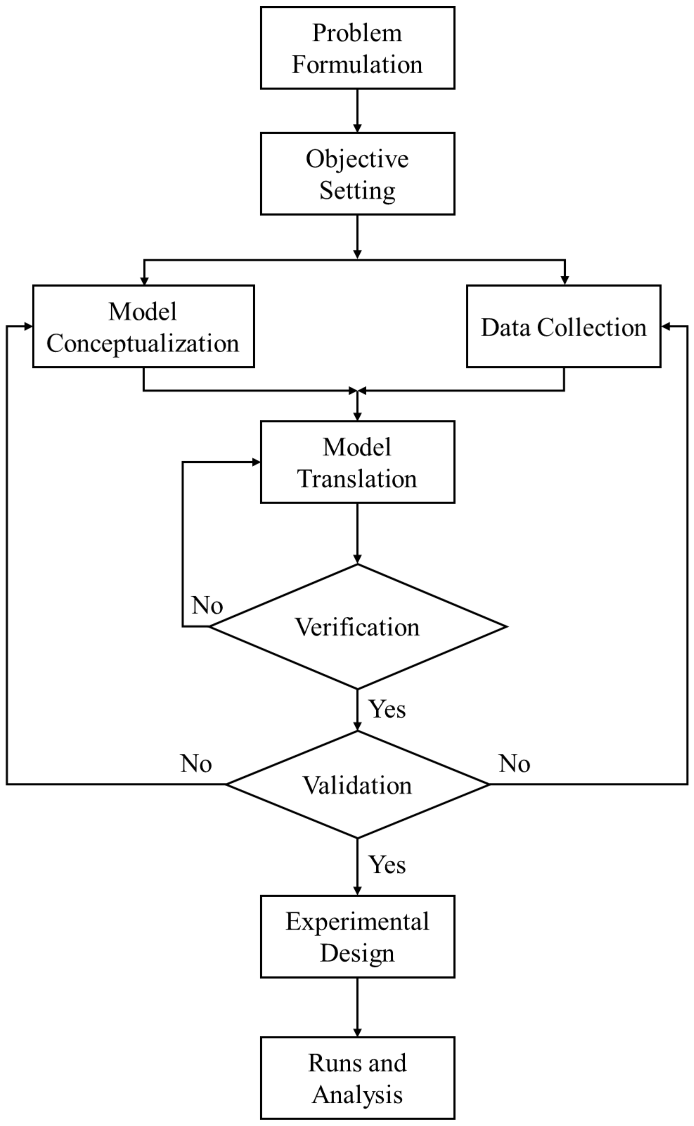


Figure 7.2 The workflow of Simulation Model Development

7.2 Performance Measurement

To evaluate the developed model, performance measurement is the critical procedure. Two essential questions that must be answered to better understanding model evaluation are why to measure and what to measure (Lebas, 1995).

The first question, why to measure, answers the usage of measures for model evaluation. The chosen measures are not the direct objectives of the designed model, but the results of a series of decisions and operations made by the model. By analyzing the measures, the comparison can be made between different models to find the key factors that influence the model operations. Also, measures provide a direction to improve the current model.

The second question, what to measure, is equivalent to asking what performance in this operational system is. Performance can be defined as the potential for future successful implementation of actions to reach the objectives (Lebas, 1995). This can be subjective, and people from different perspectives can have different answers. For example, when evaluating an elevator dispatching system, the building owner may care more about the operation and energy costs, while passengers choose the average waiting time as the most important factor. Because models with different usages of situational information will be compared in this study, the chosen performance measures should remain the same during the analysis.

7.3 Results Analysis for Optimization Calibration and Adaptability Adjustment

As discussed above, the main objective of discrete-event simulation in this study is to enhance the optimization model by adjusting the usage of situational information during optimization. To be specific, there are two methods to embed the situational information: to serve as the condition to change the problem formulation in different usage scenarios, and to add situational-related factors to the objective function. Thus, by analyzing the experiment results, the optimization model can be improved from two perspectives: the calibration of situation-related variables, and the adaptability of the optimization model to different usage scenarios.

Optimization calibration is about finding a proper value for the coefficient (the weight) of situation-related variables. Unless there is an explicit mathematical relationship, the assignment is usually intuitive. By observing the operations made during the simulation or analyzing the experiment results, the coefficient value can be changed to adjust the importance of the situational information during optimization.

There are two possible approaches to improve the model adaptability: the modification of applicable conditions for using situational information and the change of problem formulation. Model adaptability is related to the optimization strategy. And by comparing the operations made between two different models in the same experiment settings, the study can be made of whether the triggering conditions for changing the problem formulation is proper or whether the new formulation is reasonable.

Therefore, the functionality of discrete-event simulation in the proposed optimization framework is to provide feedback about simulation performance. Through results analysis and observation of the simulation process, the integration of situational information can be continuously updated until the optimization model has satisfying adaptability and proper settings for situational-related variables.

7.4 Chapter Summary

This chapter mainly introduces the development procedure of a discrete-event simulation model and the analysis of experiment results. The workflow of developing a simulation model from the real-world problem is presented. Performance measurement is discussed as the requirement for choosing model evaluation indicators. The directions for optimization enhancement through simulation results analysis are discussed: Optimization calibration is to find a proper value for the coefficient of the situation-related variable; Adaptability adjustment is to modify both the condition and the problem formulation when situational information is used to identify different scenarios.

CHAPTER 8. SIMULATION AND SYSTEM VALIDATION: AN APPLICATION OF OCCUPANCY-AWARE ELEVATOR DISPATCHING OPTIMIZATION

This chapter presents the application of dispatching optimization for elevator group control with occupancy awareness to validate CNN-enabled visual data analysis and intelligent reasoning for real-time optimization and simulation.

Elevators are the primary vertical transportation method in modern buildings, meeting the transport demands of building occupants for their work and living every day. In high-rise buildings where passengers spend much time on elevator travel during the rush hour, there are growing demands for improving elevator dispatching performance. Dispatching performance can be evaluated by the quantity of service and the quality of service (Fernandez & Cortes, 2015). The former refers to the passenger handling capacity during peak traffic periods, while the latter includes multiple indicators, such as passenger travel time, power consumption, passenger waiting time, and so on.

In general, elevator dispatching is an optimization problem to make proper hall call allocations at a real-time scale, and the main challenge is the balance between dispatching performance and its implementation costs, including computing costs, operation costs, and facility costs. An efficient dispatching system can make reasonable allocations in a short time by the preference criteria set by elevator operators. In this application, real-time occupancy awareness is proposed to extract information from inside of the elevator cab for optimal elevator group control. Chapter 8.1 introduces the optimal elevator dispatching in detail. Chapter 8.2 presents the occupancy awareness in elevator dispatching. Chapter 8.3 discusses context information extraction with the Mask R-CNN. Chapter 8.4 presents the situational information extraction with case-based reasoning and fuzzy logic. The dispatching optimization model with the prioritized A* search is introduced in Chapter 8.5. Discrete-event simulation experiments and system validation are presented in Chapter 8.6.

8.1 Optimal Elevator Dispatching

Elevator group control (EGC) refers to managing multiple elevators in a group to improve transportation efficiency and reduce operation costs (Kim et al., 1995). This is achieved by allocating hall calls to the most suitable elevator cab, in an effort to optimize one or multiple criteria, including average waiting time, average journey time, round trip time, and building owner’s preferences, like power consumption (Fernandez & Cortes, 2015; Barney & Al-Sharif, 2015). Three main problems concerned with EGC are what information to use for optimal dispatching, which dispatching optimization algorithm to apply, and how to determine the traffic pattern or the usage pattern to deploy different dispatching strategies.

Dispatching information is a set of parameters describing elevator operating states and is applied as the input of dispatching optimization problems. Conventional dispatching information includes the cab position and direction, existing hall calls, registered cab calls, and so on (Al-Sharif, 2016). The more useful information used during dispatching, the better dispatching performance may be obtained, though that often requires more hardware support to capture the needed data.

Dispatching optimization aims at finding the most proper hall call allocations that optimize the selected criteria with the given dispatching information. Decision making for elevator dispatching is essentially a combinatorial optimization problem and falls into the NP-complete problems (Garey & Johnson, 1979). Given the characteristics of NP-complete problems, prevailing optimization algorithms are heuristics-based, because they can usually find a near-optimal solution in a short time. Other algorithms like fuzzy logic, neural networks are also used (Fernandez & Cortes, 2015; Hamdi & Mulvaney, 2007).

Dispatching strategies regulate elevator operations in different scenarios, including the control strategies under different traffic patterns, sub-zoning, and sectoring. Based on the analysis and prediction of traffic flow, traffic patterns can be roughly clustered into three: up-peak, down-peak, and inter-floor (Siikonen, 1997). Then, different control strategies are deployed to fit the characteristic of the current traffic pattern. Zoning and sectoring techniques are developed to specify working regions for different cabs (Chan et al., 1996; Li et al., 2007). To reduce the number of stops and the journey time, zoning divides the building with floors, like a lower zone and an upper zone. During the current traffic pattern, each elevator only serves one zone. On the other hand, sectoring divides the building in relation to both the position and direction. Such division can change according to the traffic flow, and each sector is assigned to only one cab (Al-Sharif, 2016).

8.2 Elevator Occupancy Awareness

As one common criterion for dispatching optimization, the journey time is defined as the time interval between a passenger registers a hall call and the passenger arrives at the destination floor (Bloat & Cortes 2011), which is the sum of waiting time and traveling time. The number of stops in a trip can be a significant factor influencing the journey time. Because the movement speed of each elevator cab is relatively fixed, executing more stop operations to reach a floor means more time is spent on door closing and opening. This time not only influences the traveling time of the passengers in the cab, but also increases the waiting time of passengers waiting for this cab. Considering there are scenarios where a hall call fails to pick up any passenger due to the cab capacity or where passengers are not supposed to use the elevator when it is in a special usage, this application introduces occupancy information to improve the dispatching performance under such circumstances.

8.2.1 Occupancy Information Analysis

When a cab has no capacity for new passengers but is not overweighed, or when it is in a special usage and cannot receive new hall calls (like in hospitals), prevailing dispatching systems would keep assigning hall call tasks to it, thus generating wasted stops and increasing the journey time. Sometimes, lack of capacity is not because the cab is fully occupied but caused by large obstacles that block the entrance of the cab, like when passengers bring bicycles and place them at the entrance. And an example of a cab in special usage is the transfer of a patient on a stretcher that needs first-aid, and the cab should not answer new hall calls until the existing task is performed.

The hidden assumption of traditional elevator dispatching systems is there is enough capacity to accommodate new passengers, so the dispatching mainly focuses on time-based criteria, like estimated time of arrival. However, capacity is a type of resource during dispatching, and if the resource constraint is not satisfied, penalties should be imposed on passenger waiting time and travel time.

To improve the current dispatching system, occupancy awareness is proposed during optimization. One objective is to reflect the cab capacity, thus decreasing the probability of pick-up failure and improving dispatching performance. Also, objects in the environment can be detected to recognize the current usage. And the number of passengers can be used for traffic analysis. The optimization information hierarchy is shown in Figure 8.1.

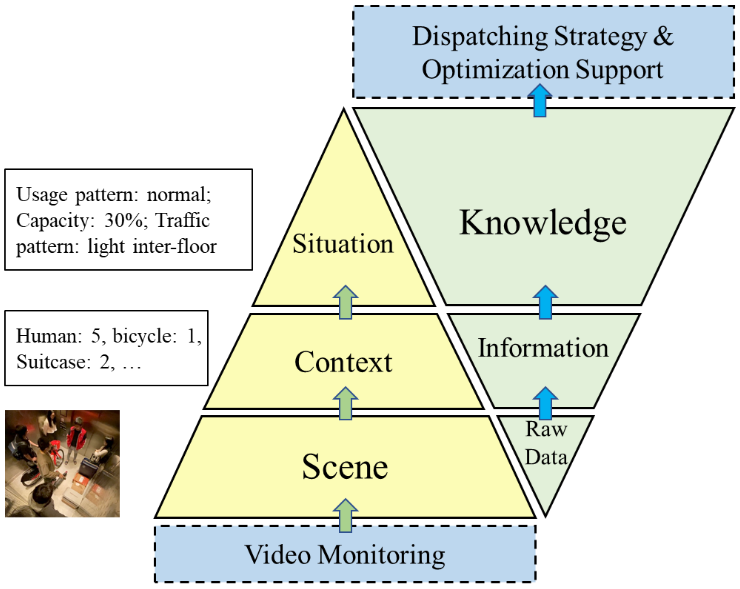
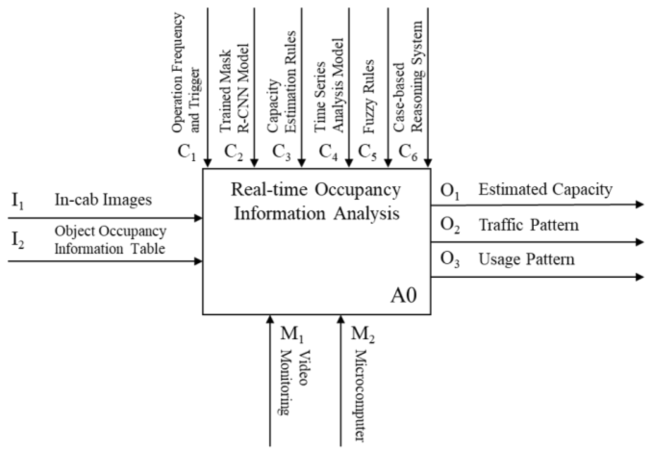

Dispatching Strategy & Optimization Support
Usage pattern: normal; Capacity: 30%; Traffic pattern: light inter-floor
Situation
Knowledge
Human: 5, bicycle: 1, Suitcase: 2, …
Context
Information
Scene
Raw Data
Video Monitoring

Figure 8.1 Occupancy information hierarchy

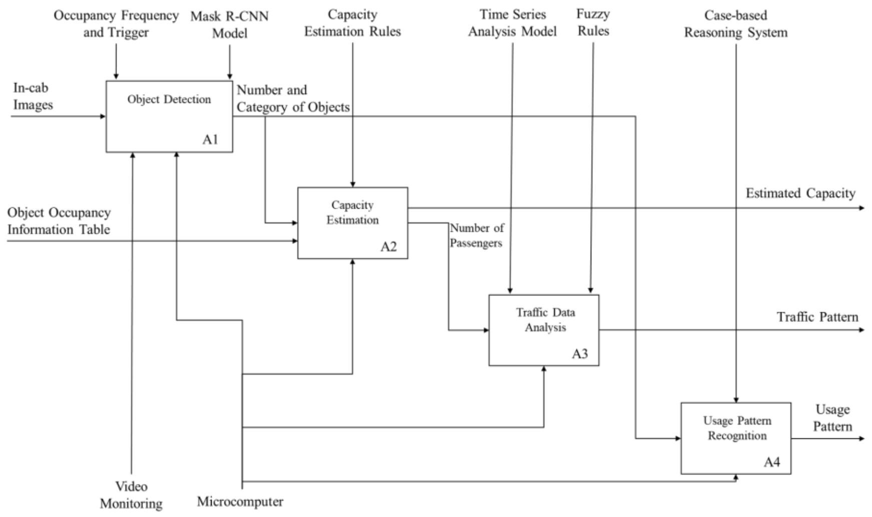
The first level is the hardware level, collecting video data inside the elevator cab with video monitoring. This level aims at recording the objects in the scene and provides image data for further occupancy analysis. The second level is to extract context information. Object detection is implemented to aggregate the number and category of the objects in the cab with Mask R-CNN. The number of passengers in each cab during operation is recorded separately. The context information is used for traffic pattern recognition and occupancy analysis. The third level is to extract situational information. This level is to refine the obtained information as the capacity factor to avoid pick-up failure and recognize the current elevator usage and the traffic pattern. Specifically, the cab capacity is estimated with space utilization and the mobility of objects in the cab. And the usage is recognized with case-based reasoning to determine if any specific dispatching strategy should be applied. The traffic pattern can be recognized through fuzzy logic using the real-time data of the number of passengers.

8.2.2 Functional Analysis

The occupancy information analysis model is the model that implements real-time occupancy awareness for optimal elevator dispatching. The functional analysis diagram (IDEF0) of the model is shown below. Figure 8.2(a) is the overview of the model, while Figure 8.2(b) presents its details. The whole process is completed on a microcomputer (M2). Firstly, object detection (A1) is performed with a pre-trained Mask R-CNN model (C2) using in-cab image data (I1) collected from the video monitoring (M1). The detection is executed with the predefined frequency or triggers (C1) to reach a real-time scale and to detect transferred passengers during a stop. The detection result that aggregates the numbers and categories of objects is used for situational awareness, including capacity estimation (A2) and usage pattern recognition (A3). Capacity is estimated based on the occupancy area contributed by each object and their mobility, which are predefined and recorded in the object occupancy information table (I2). The specific prediction method is to model the “consistency level” of the 3D space with a set of predefined rules (C3). And the usage pattern is determined by searching target objects in the detection result and finds the corresponding dispatching strategy with case-based reasoning (C6). Suppose a stretcher is detected in a hospital elevator cab. In that case, the cab will be determined as in the emergency usage, and the system will hold the hall call and cab call tasks of the cab and stop assign new tasks to the cab until the stretcher leaves, since a patient that needs first-aid should be served as a priority. Traffic data analysis (A3) is to analyze the passenger arrival data in different traffic and recognize the current traffic pattern, which requires the time series analysis model (C4) for passenger arrival rate forecast and fuzzy rules (C5) to analyze the traffic intensity and traffic components.



(a) Overview of the functional analysis of the model.



(b) Detailed functional analysis of the model.

Figure 8.2 Functional analysis of the occupancy information analysis model

8.3 Real-time Context Information Capturing with Mask R-CNN

In the proposed elevator dispatching system, context awareness aims to find the object type and number in the cab. Thus, Mask R-CNN is applied to implement object detection for real-time context information capturing.

8.3.1 Data Set Construction

Dataset construction is an important step in model development. Because most detection targets in the cab are everyday objects, like humans, backpacks, and suitcases, an existing dataset can be used to save time from image collection and labeling. In this application, the Microsoft COCO dataset is used for the training, which contains 320,000 labeled images and 90 object categories for object detection (Lin et al., 2014). As cardboard boxes are frequently seen in an elevator, box images are also collected and labeled as a new class in the dataset. LabelMe is used to annotate new data. For the convenience of training and testing, the image size is scaled to 640 * 480. The final dataset consists of cardboard boxes and 13 categories of objects from the COCO dataset: backpack, bicycle, cat, chair, dog, handbag, person, snowboard, sports ball, suitcase, surfboard, tennis racket, and umbrella. 2780 images are selected, where 1668 are in the training set and 556 for validation, and the remaining 556 images are in the testing set.

8.3.2 Model Training

The model in this application is trained using transfer learning techniques, which saves the time by learning the extraction of general features from pre-trained models (Shin et al. 2016). This is implemented by firstly freezing the backbone layers and only training the head layers. Then the entire model is fine-tuned. The training is done on the new dataset. Normally, ResNet-50 is chosen for backbone feature extraction.

An example of object detection in an elevator cab is presented in Figure 8.3.

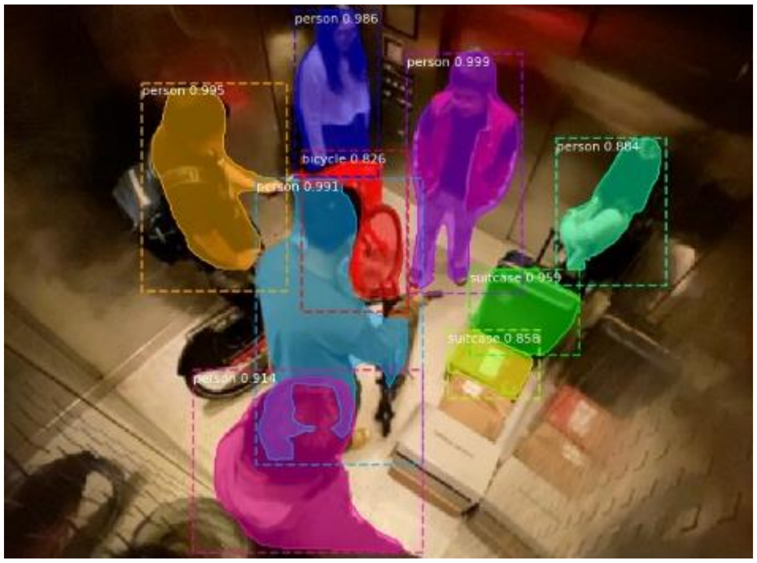


Figure 8.3 Object detection in the elevator cab

8.4 Case-based Reasoning and Fuzzy Logic for Elevator Situation Comprehension

8.4.1 Situation Comprehension during Elevator Dispatching

To improve dispatching performance, situational information is extracted to equip the optimization model with the comprehension of the current environment. In elevator dispatching, three types of situational information are useful for obtaining high-quality dispatching solutions: the cab usage pattern, the traffic pattern, and the estimated capacity.

The objective of usage pattern recognition is to recognize the transport objective of the trip. If the cab is in a special usage, it may perform only the specified tasks, and this can change the number of usable cabs or the hall calls that should be answered in the optimization model. Traffic patterns indicate the traffic components about passengers' trip directions and the traffic intensity. If the traffic is heavy and most passengers have the same trip directions, like the rush hour in the morning, certain dispatching strategies can be adopted to make more reasonable hall call allocations. Capacity estimation is to understand the space utilization of each cab. It enables the optimization model to consider the capacity constraints during dispatching in heavy traffic, so the hall call allocation can be more balanced to avoid pick-up failure caused by the lack of capacity.

8.4.2 Case-based Reasoning for Elevator Usage Pattern Recognition

Because elevators can be in special usages during operations, timely detection of a special usage can help the dispatching system to adjust its dispatching strategy and bring convenience to passengers. For example, if an elevator cab is often used to transport freight, a usage pattern regarding this usage can be set. If specific freight carriers or devices are detected, the usage is defined as a special one. In this scenario, a special optimization strategy can be set, like to serve the current task as a priority and ignore the outside hall calls until the current task is finished. Different dispatching strategies can be set to satisfy the demands of special elevator usages from the management perspective.

In this application, case-based reasoning is used to implement usage pattern recognition. There are five steps: case representation and indexing, new case creation, case retrieval, case reuse or case adaptation, and case retention.

Case representation and indexing are the preparation work for further operations. Because the objective is to detect special usages, the number and type of key items can be used to describe a case. For example, stretchers and cleaning carts are set as the target object for the first-aid usage and cleaning tool transport. And the number of passengers is also of interest. Then $< c_1, c_2, c_3, ID, ans >$ can be used to describe a case, where c_1 and c_2 are Boolean variables (true or false) representing the existence of stretchers and cleaning carts, and c_3 is a numeric variable representing the number of passengers in the cab. ID is the index of the case, and ans is its recorded solution. The case indexing can use the special usage as the case category and a number that represents the sequence of that case in this category to ease the case retrieval process. Table 8.1 presents two cases for the transport of a cleaning cart and the freight.

Table 8.1 Two examples of case representation

(a) Cleaning cart transport

-ID: Cleaning_003 -CleaningCart: True -Freight: False -Psg_Number: 4 -Capacity: 30% -Direction: upward
+Solution: The cab becomes unavailable for new calls till the cart leaves.

(b) Freight transport

-ID: FreightTransport_015 -CleaningCart: False -Freight: True -Psg_Number: 0 -Capacity: 30% -Direction: downward
+Solution: Clear current hall call allocations and the cab become unavailable for new calls till the freight leaves.

Case retrieval aims to find similar cases in the case library, and similarity measuring between the new case and old cases is needed. A conventional approach is to use the nearest neighbor matching with the following equation for similarity measuring:

$$\frac{\sum_{i=1}^{n} w_i * similarity(c_i^{new}, c_i^{old})}{\sum_{i=1}^{n} w_i} \tag{8.1}$$

where c_i is the case feature, and w_i is the weight of the feature. Because the usage detection is largely determined by the existence of items, the corresponding feature variable is either Boolean or integral, making the similarity measuring easier.

The objective of case adaptation or reuse is to find a solution to the current case. Based on different elevator usages and the detected items, there can be several types of solutions. For example, if the elevator is used for first aid, the current hall call and cab call assignment will be eliminated to serve the current task, and no new hall call will be allocated until this task is performed. Suppose the elevator is used for the transport of a cleaning cart. In that case, the cab will keep its current cab call assignment, but whether to execute existing hall calls can be decided by other information, like the floor difference between the cart destination and its starting floor. Production (IF-THEN) rules can be used to evaluate the case similarity and decide whether to modify or reuse the original solution. After the solution is found, the solved new case can be stored in the case library for future reference.

8.4.3 Traffic Pattern Recognition with Fuzzy Logic

8.4.3.1 <u>Passenger Travel Data Analysis</u>

According to the travel purpose, passenger traffic components can be categorized into incoming traffic, inter-floor traffic, and outgoing traffic. To determine the passenger arrival rates in different traffic, the data recording the number of passengers in each cab is transformed into the number of passengers in different traffic.

Context awareness provides the real-time data of the number of passengers. Considering that passengers only enter or exit the cab during a stop, a frequency can be determined to capture the change during the stop. Assume passengers do not enter the cab until all the passengers whose destination is at this floor exit. Let n_{p0}, n_{pmin}, n_{p1} be the number of passengers before the stop, the minimum number of passengers during the stop, and the number of passengers after the stop. The number of passengers exiting the cab is $n_{p0} - n_{pmin}$, while the number of new passengers is $n_{p1} - n_{pmin}$.

To derive the passenger arrival rates in different traffic, the number of transferred passengers is needed. For incoming traffic, the number of passengers who enter the cab on the main floor is recorded. For inter-floor traffic, if the cab moves upwards, the number of passengers entering the cab between the second floor and the next highest floor is recorded. If the cab moves downwards, the number of passengers who exit the cab on these floors are recorded. For outgoing traffic, the number of passengers who exit the cab on the main floor is recorded. The recorded data is then aggregated with a fixed granularity to calculate the passenger arrival rates in different traffic.

8.4.3.2 Traffic Flow Forecast with ARRES

Traffic pattern recognition aims to detect the pattern change in advance to avoid the form of long waiting queues. Thus, traffic flow forecast is implemented to predict the passenger arrival rates. In this application, the adaptive response rate exponential smoothing (ARRES) is chosen as the time series analysis model, which was validated to solve this problem (Makridakis et al., 1983). The learning of traffic flow is not limited to a single day, and the past data will be incorporated into the long-term statistics.

Compared to single exponential smoothing, ARRES continuously optimizes the smoothing factor α. The prediction formula is illustrated below.

$$F_{t+1} = \alpha_t Y_t + (1 - \alpha_t) F_t \tag{8.2}$$

$$\alpha_t = \left| \frac{E_t}{AE_t} \right| \tag{8.3}$$

$$E_t = \beta e_t + (1 - \beta) E_{t-1}, 0 < \beta < 1 \tag{8.4}$$

$$AE_t = \beta |e_t| + (1 - \beta) AE_{t-1} \tag{8.5}$$

$$e_t = Y_t - F_{t-1} \tag{8.6}$$

F_{t+1} is the forecast smoothed value, Y_t is the observation in the current period, E_t is the smoothed average error, and AE_t is the smoothed absolute error. α_t is the smoothing factor that keeps being optimized. β is a predefined constant parameter between zero and one. And e is the error term.

After the forecast, the future passenger arrival rates in different traffic, $\lambda_{inc}, \lambda_{out}, \lambda_{int}$, can be obtained.

8.4.3.3 Traffic Pattern Recognition with Fuzzy Logic

In this application, traffic pattern recognition is conducted with fuzzy logic according to the traffic intensity and the distribution of different traffic components in the building. The implementation is based on published research of KONE (Siikonen, 1997). The proposed method defines four traffic factors related to different traffic components and the overall traffic intensity. Fuzzy set values are obtained with two groups of membership functions for traffic components and the traffic intensity. Fuzzy rules are then applied to determine the traffic pattern based on the fuzzy set values.

Traffic component analysis and traffic intensity evaluation are first implemented, with four traffic factors, u_1, u_2, u_3, u_4, defined to grade the corresponding level. The component analysis aims to reflect the proportion of three traffic. The component values for incoming traffic, outgoing traffic, and inter-floor traffic, u_1, u_2, u_3, are shown in Equation (8.7)-(8.9).

$$u_1 = 100 * \lambda_{inc}/(\lambda_{inc} + \lambda_{out} + \lambda_{int}) \tag{8.7}$$

$$u_2 = 100 * \lambda_{out}/(\lambda_{inc} + \lambda_{out} + \lambda_{int}) \tag{8.8}$$

$$u_3 = 100 * \lambda_{int}/(\lambda_{inc} + \lambda_{out} + \lambda_{int}) \tag{8.9}$$

And the overall traffic intensity value u_4 is to scale the total arrival rates to the passenger handling capacity HC, where the handling capacity is determined as the number of served passengers in five minutes during up-peak traffic:

$$u_4 = 100 * (\lambda_{inc} + \lambda_{out} + \lambda_{int})/HC \tag{8.10}$$

Then fuzzy sets {low, high, medium}$\in \mathcal{F}_1$ are used to describe u_1, u_2, u_3. Figure 8.4 presents the fuzzy sets corresponding to traffic components. The membership function to describe the three factors are as below:

$$\mu_{low}(u) = \begin{cases} 1, if\ u < 25 \\ \frac{35-u}{10}, if\ 25\ \le u < 35 \\ 0, if\ u \ge 35 \end{cases} \tag{8.11}$$

$$\mu_{medium}(u) = \begin{cases} 0, if\ u < 30 \\ \frac{u-30}{20}, if\ 30 \le u < 50 \\ \frac{70-u}{20}, if\ 50 \le u < 70 \\ 0, if\ u \ge 70 \end{cases} \tag{8.12}$$

$$\mu_{high}(u) = \begin{cases} 0, if\ u < 65 \\ \frac{u-65}{10}, if\ 65 \le u < 75 \\ 1, if\ u \ge 75 \end{cases} \tag{8.13}$$

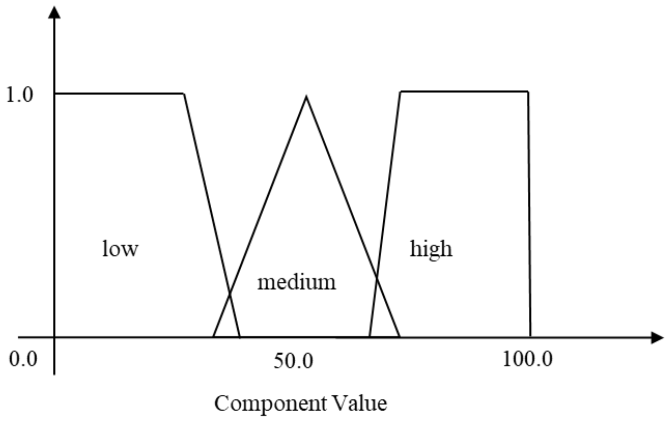


Figure 8.4 Membership functions for different traffic components

The obtained results are used to determine the traffic type with the fuzzy rules presented in Table 8.2. The traffic type will influence the parameter setting in the membership function for the traffic intensity.

Table 8.2 Fuzzy rules to determine the traffic type (Siikonen, 1997)

Incoming	Outgoing	Inter-floor	Traffic Type
high	low	low	incoming
medium	low	low	incoming
low	high	low	outgoing
low	medium	low	outgoing
low	low	high	inter-floor
low	low	medium	inter-floor
medium	medium	low	two-way
medium	low	medium	mixed
low	medium	medium	mixed

The fuzzy set for the traffic intensity is {light, normal, heavy, intense}$\in \mathcal{F}_2$, and the membership functions are shown in Figure 8.5. The formulae are presented in Equation (8.14).

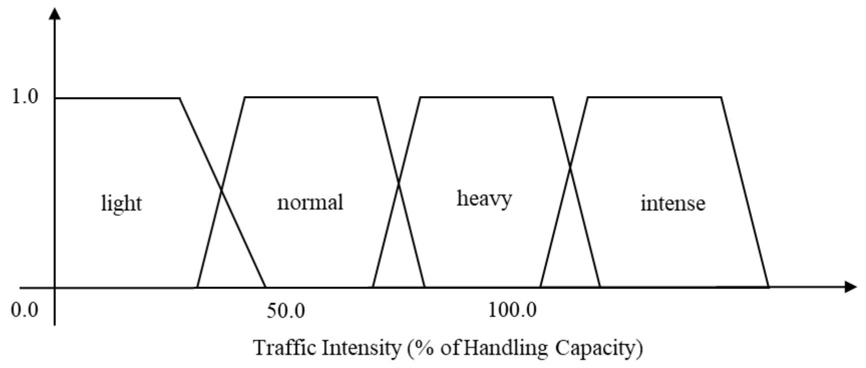


Figure 8.5 Membership functions for the traffic intensity

$$\mu_f(u_4) = \begin{cases} 0, if\ u_4 < a_j \\ \dfrac{u_4 - a_j}{b_j - a_j}, if\ a_j \le u_4 < b_j \\ 1, if\ b_j \le u_4 < c_j \\ \dfrac{d_{fj} - u_4}{d_j - c_j}, if\ c_j \le u_4 < d_j \\ 0, if\ u_4 \le d_j \end{cases} \tag{8.14}$$

where $a_j \le b_j \le c_j \le d_j, j \in J$. J is the space of all traffic types. Different set limits, a_j, b_j, c_j, d_j, are set according to the traffic type.

Finally, the grades of the membership function for traffic components and the traffic intensity are compared with the equation below:

$$\begin{aligned} \mu_{i'}(u_1, u_2, u_3, u_4) &= \mu_i(u_1) \wedge\ \mu_i(u_2) \wedge\ \mu_i(u_3) \wedge \mu_f(u_4) \\ &= \min\{\mu_i(u_1), \mu_i(u_2), \mu_i(u_3), \mu_f(u_4)\} \end{aligned} \tag{8.15}$$

where $i \in \mathcal{F}_1, f \in \mathcal{F}_2, i' \in Z$. Z is the space of all traffic patterns.

Table 8.3 presents the 36 fuzzy rules to determine traffic patterns. The grades obtained with Equation (8.15) are added to the corresponding rules. The rule with the highest grade determines the recognized traffic pattern.

Table 8.3 Fuzzy rules for traffic pattern recognition (Siikonen, 1997)

Intensity	Incoming	Outgoing	Inter-floor	Traffic Pattern
intense	high	low	low	intense up-peak
intense	low	high	low	intense down-peak
intense	low	low	high	intense inter-floor
intense	medium	low	low	intense up-peak
intense	low	medium	low	intense down-peak
intense	low	low	medium	intense inter-floor

intense	medium	medium	low	intense mixed
intense	medium	low	medium	intense mixed
intense	low	medium	medium	intense mixed
heavy	high	low	low	up-peak
heavy	low	high	low	down-peak
heavy	low	low	high	heavy inter-floor
heavy	medium	low	low	heavy up-peak
heavy	low	medium	low	heavy down-peak
heavy	low	low	medium	heavy inter-floor
heavy	medium	medium	low	mixed
heavy	medium	low	medium	heavy mixed
heavy	low	medium	medium	heavy mixed
normal	high	low	low	up-peak
normal	low	high	low	down-peak
normal	low	low	high	inter-floor
normal	medium	low	low	up-peak
normal	low	medium	low	down-peak
normal	low	low	medium	inter-floor
normal	medium	medium	low	mixed
normal	medium	low	medium	mixed
normal	low	medium	medium	mixed
light	high	low	low	light up-peak
light	low	high	low	light down-peak
light	low	low	high	light inter-floor
light	medium	low	low	light up-peak
light	low	medium	low	light down-peak
light	low	low	medium	light inter-floor
light	medium	medium	low	light mixed
light	medium	low	medium	light mixed
light	low	medium	medium	light mixed

8.4.4 3D-Space Consistency Modelling with Object Mobility for Capacity Estimation

Situational awareness is proposed to understand the latent story of the cab usage. In this sense, capacity is defined as the area that can accommodate new passengers. This concept is different from the unoccupied space, because sometimes passengers cannot find a path to the unoccupied space when large obstacles are getting in the way. Considering the influence of object mobility on the capacity, this study proposes to utilize the object occupancy area, S, and the object mobility, M, to model the 3D space consistency level, L, and then to do the estimation.

The object occupancy area is the area that the object occupies and is obtained based on the crowd density approximation method by fully occupying the cab with the objects (Barney & Al-Sharif 2015). For example, the ground of a cab is measured as $2m \times 2m$. Figure 8.6 shows the static crowd density from 1 person$/m^2$ to 5 people$/m^2$. According to the crowd density and risk analysis control, 5 people$/m^2$ is the upper limit for standing spaces. Therefore, the maximum capacity for humans is 20, and the object occupancy area is 0.2 sqm.

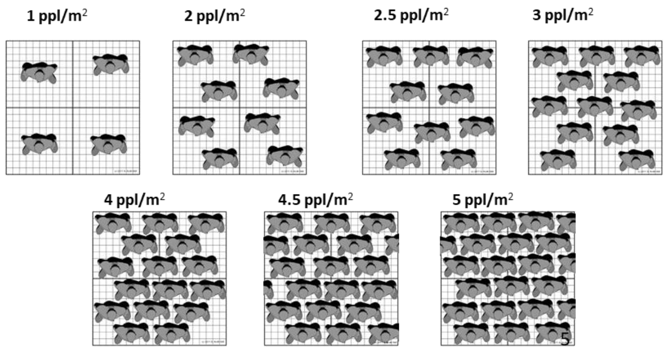


Figure 8.6 Crowd density estimation.

The object mobility is defined as a categorical variable to describe how easily the object can be moved. It has three levels: low, medium, and high. If an object is too large or heavy to move, or it is not supposed to move much in the cab, its mobility is defined as low. For example, the mobility of a wheelchair is low, because crowdedness should be avoided when the disabled are in the cab.

The 3D-space consistency level is defined to describe the ability of the cab to make room for new passengers, which can be regarded as the evaluation for conversion efficiency between the space and the capacity. The value of the consistency level is also supposed to consider the object position. For example, if all the objects in the cab have high mobility, it is easier to reach a high space utilization rate, because the positions of passengers and other objects can be adjusted. But if an object with low mobility is placed near the cab entrance, it will be inconvenient for passengers to enter. However, given that the relationship among capacity, object mobility, and position is ambiguous to model, and that the camera angle influences the object position obtained from Mask R-CNN, this study uses the mobility factor to reflect the effect of object position on capacity. Table 8.4 presents part of the object occupancy information in this application.

Table 8.4 Object occupancy information

Object	Mobility	Occupancy Area	Occupancy Percentage
Human	High	0.2 sqm	5%
Backpack	High	0.02 sqm	0.5%
Suitcase	High	0.2 sqm	5%
Scooter	Medium	0.214 sqm	6%
Box	Medium	0.3 sqm	7.5%
Wheelchair	Low	1 sqm	15%
Bicycle	Low	0.5 sqm	12.5%

In this application, the mobility value M corresponds to the predefined mobility level:

$$M = \begin{cases} 1, \text{level} = \text{high} \\ 2, \text{level} = \text{medium} \\ 3, \text{level} = \text{low} \end{cases} \tag{8.16}$$

The value of room consistency level is calculated as a function of the object occupancy area S and the object mobility M:

$$L = \frac{\sum_{i=1}^{N} S_i M_i}{\sum_{i=1}^{N} S_i} \tag{8.17}$$

where N is the number of detected objects.

The estimated capacity C is calculated as the empty space divided by the room consistency level:

$$C = \frac{1}{L} \cdot \left(S_{\text{cab}} - \sum_{i=1}^{N} S_i\right) \tag{8.18}$$

where S_{cab} is the ground area of the cab.

8.5 Real-time Dispatching Optimization with Prioritized A* Search

The elevator dispatching problem falls into NP-complete problems and requires real-time responses. In a building with n elevators with p hall calls, the number of possible solutions is n^p. During peak traffic, the dispatching system needs to make hall call allocations in a short time with high quality. In this application, a modified prioritized A* search algorithm is used to find the optimal allocations considering occupancy information (Hamdi & Mulvaney, 2007).

8.5.1 Prioritized A Search Algorithm*

Compared to the heuristics-based methods, the modified prioritized A* search is more suitable for this context. Firstly, prevailing heuristics will not stop until the convergence criteria are met. However, for elevator dispatching, hall call allocations should be assigned within very few seconds after the hall call is registered. Secondly, heuristics usually find a general high-quality solution and have a thorough search horizon. However, dispatching information keeps changing all the time, and original allocations are less meaningless since new hall calls are generated. Thus, it is advantageous if the algorithm can limit the search horizon and interrupt the search horizon. These operations are easy to implement for the A* search algorithm. Thirdly, the cost of each hall call allocation is reflected in the objective function of the A* search algorithm, making it easier to fit the occupancy information into the problem.

In the context of elevator dispatching, each level addresses one hall call assignment in the A* search algorithm. When the number of hall calls exceeds the number of cabs, this algorithm tends to assign all hall calls until it terminates. However, in this scenario, it is more important that each cab has at least one hall call task. If the algorithm is interrupted because of the limit of the computing time, there might be cabs with no hall call assignment.

In this regard, the prioritized A* search is proposed and prioritizes the search sequence according to the elevator usage. This means the objective of the search is to allocate a hall call task to a cab at each tree level. Because the search logic forces the assignment of cabs to hall calls instead of the reverse, the search can be stopped when the number of the searched tree levels is equal to the number of cabs.

8.5.2 Dispatching Optimization Problem Formulation

To introduce the occupancy information into the problem formulation, the capacity C_n and cab ground area S_n are utilized, where n refers to the cab sequence. Based on the influence of the capacity on pick-up failure, the cost to perform a hall call task should be smaller if a cab has more capacity, and vice versa. Thus, the occupancy factor o_n is set as the coefficient of the task cost, and it is calculated as below:

$$o_n = \begin{cases} \sqrt{\frac{S_n}{C_n}}, if\ C_n > 0.01S_n \\ 10, if\ C_n \leq 0.01S_n \end{cases} \tag{8.19}$$

It should be noticed that the occupancy factor o_n should only be applied to the hall call tasks that are performed before the cab changes the direction. Because the assumption of the elevator operation logic is that the cab will not change its direction before it finishes all the tasks in the original direction. If the hall call task is performed after the cab changes the direction, passengers on the cab currently will exit the cab before the last task in this direction, and the occupancy factor will be no longer effective to the task costs.

The passenger waiting time is chosen as the criterion to evaluate dispatching performance. Correspondingly, the calculation of g and h are concerned with the waiting time. Two look-up tables are prepared to calculate the value g and h. The first table records the costs for each cab to answer a hall call as its first hall call from the current position, taking the cab call commitments into account. And the second table records the costs of trips between different pairs of hall calls for each cab, in which both hall call and cab call tasks are considered. The cost estimation is based on the worst-case scenarios to ensure no overestimation is made on h. Let d_n be the cost to answer a hall call as its first hall call task for the cab n, and s_n be the cost of a trip between a pair of hall calls. Let p_i be an assigned hall call of the cab n, and P_n be the number of assigned hall calls to it. Thus, the first table represents $d_n(p_1)$, referring to the cost for the cab n to answer the hall call as the first hall call task. The second table represents $s_n(p_i, p_{i+1})$, meaning the cost for the cab n to answer the hall call p_i followed by p_{i+1}. For each hall call m performed before the cab changes its direction, the cost $g_n(m)$ is given by:

$$g_n(1) = o_n * d_n(p_1), \quad P_n = 1 \tag{8.20}$$

$$g_n(m) = o_n * d_n(p_1) + \sum_{i=1}^{P_n-1} o_n^{\frac{1}{3i}} s_n(p_i, p_{i+1}), \quad P_n > 1, m = 1, \dots, P_n \tag{8.21}$$

On the other hand, if the hall call m is performed after the cab n changes the direction, let v_1 be the number of hall call tasks before it changes the direction, the cost $g_n(m)$ is given by:

$$g_n(1) = d_n(p_1), \quad P_n = 1 \tag{8.22}$$

$$g_n(m) = o_n * d_n(p_1) + \sum_{i=1}^{v_1-1} o_n^{\frac{1}{3i}} s_n(p_i, p_{i+1}) + \sum_{j=v_1}^{P_n-1} s_n\left(p_j, p_{j+1}\right), \; P_n > 1, m = 1, \dots, P_n \tag{8.23}$$

The value of g at the current node is the sum of g_n. It should be noted that the occupancy influence on the cost evaluation of performing a task will decrease dramatically as the cab receives more tasks. The same goes for the calculation of h.

Let Q be the number of hall calls to be assigned and q_j is one of the hall calls, and k be an unassigned hall call temporarily assigned at the current node. Usually, Q is chosen as the number of the cabs in the system. The cost $h_n(k)$, for the cab n to answer a hall call q_j as the k_{th} assigned hall call performed before the cab changes the direction, is given by:

$$h_n(k) = o_n * \min[d_n(q_j)]\,, P_n = 0, j = 1, \dots, Q \tag{8.24}$$

$$h_n(k) = \min\left[\sum_{m=1}^{P_n} g_n(m) + o_n^{\frac{1}{3P_n}} s_n\left(p_{P_n,q_j}\right)\right], P_n > 0, j = 1, \dots, Q \tag{8.25}$$

If this hall call is performed after the cab n changes its direction, the cost $h_n(k)$ is given by:

$$h_n(k) = \min[d_n(q_j)]\,, P_n = 0, j = 1, \dots, Q \tag{8.26}$$

$$h_n(k) = \min\left[\sum_{m=1}^{P_n} g_n(m) + s_n\left(p_{P_n,q_j}\right)\right], P_n > 0, j = 1, \dots, Q \tag{8.27}$$

The value of $h(k)$ is the smallest value from $h_n(k)$, and the value of h is the sum of $h(k)$. In the meantime, the value h is used as the termination criterion. If $h = 0$ at a node, that node will be the goal state.

8.6 Simulation Experiments and System Validation

To validate the proposed smart dispatching with real-time occupancy awareness, the simulation model for elevator dispatching is developed, and discrete-event simulation experiments are conducted. To study the effects of dispatching with occupancy information and simplify the simulation process, only capacity estimation is implemented, meaning that the existence of objects is known and recorded once they are generated, and no special usage pattern is set during the experiments. Furthermore, objects in the experiments are designed to have relatively large occupancy area but small weight.

8.6.1 Simulation Model Development

Simulation for elevator dispatching requires three components: one for building the discrete-event simulation model, one for implementing the dispatching optimization algorithm, and one for data communication between the simulation model and the algorithm. In this study, Simio is the simulation software to model the behavior of elevator cabs and passengers. MATLAB is used for computing work, including capacity estimation and dispatching optimization. And Microsoft SQL Server is the data communication platform. A user-defined Simio process is developed with C# based on the Simio API to call MATLAB function from Simio, so that Simio can issue the command to run an algorithm in MATLAB during the simulation.

Some assumptions are made based on the elevator usage in real life: (i). Passengers cannot enter the cab if it does not have enough capacity or they make the cab overweighed. (ii). If passengers cannot enter the cab when it arrives, they will repress the hall call button after it leaves. (iii). If the cab is fully occupied, it still stops at the floor where its allocated hall call tasks are registered. (iv). Passengers only enter the cab that goes towards their destinations. (v). Passengers arrive one by one, instead of in a group. (vi). A cab will park on the ground floor if it has no tasks. (vii). A cab will not change the direction until all the hall call and cab call tasks are fulfilled in the original direction. (viii). The time to travel to the adjacent floor is fixed. (ix). The time is the same to perform either a hall call task, a cab call task, or both on a floor.

In Simio, the basic object unit is the “entity”. Built-in entities like vehicles and workers in the standard library have specific properties, processes, and functions to perform certain tasks. Considering the complexity of the elevator operation logic, this study develops the elevator and passenger objects from scratch to model their behaviors instead of using built-in entities. Apart from this, some necessary state variables are defined as the model states for the convenience of data communication.

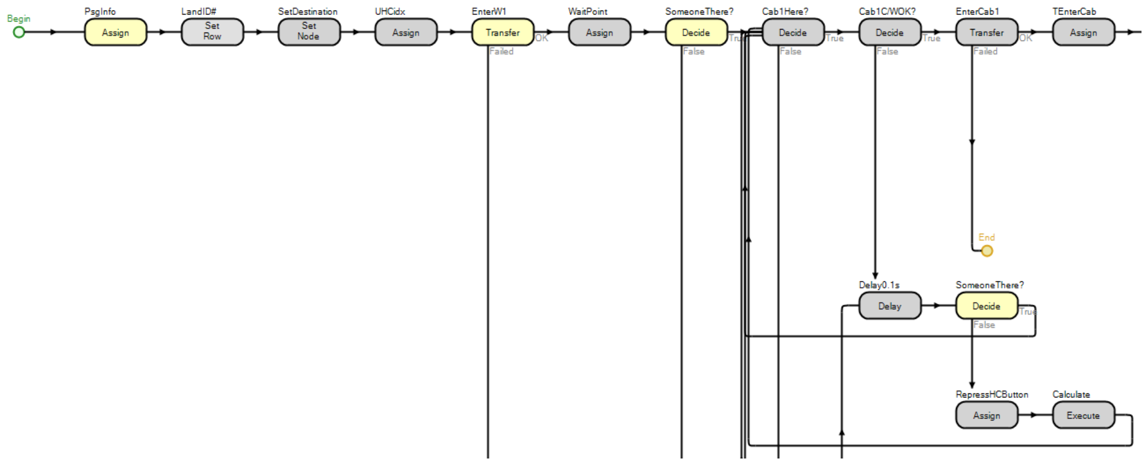
8.6.1.1 Model State Variable Setup

To guarantee the accessibility of some key variables in the process of any entity object, they are defined as the model state variables. These include the weight and capacity of each elevator cab, arrays that record the allocated hall calls and the committed cab calls of each cab, and an array recording the unassigned hall calls. Meanwhile, total passenger waiting time, total passenger journey time, and the number of the served passengers are defined.

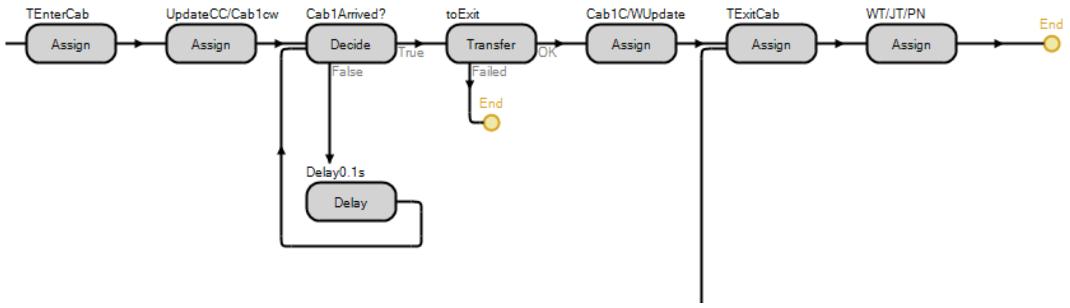
8.6.1.2 Passenger Object Development

Passenger processes can be grouped according to whether the passenger enters a cab or not. When the passenger is waiting, the process ensures that the passenger's information is initialized, and the unassigned hall call array is updated. After the passenger enters the cab, the process is to update the cab variables and check if they arrive at the destination floor.

The key process is shown in Figure 8.7. After a passenger is generated from the source, related state variables are initialized, including traffic data like the birth floor and destination floor and data for calculating the weight and cab capacity. The passenger is then transferred to the waiting area of that floor, and the unassigned hall call array is updated in the corresponding index. After the preparation, there are conditional statements to check if any cab arrives at the floor and if the cab can be taken. If so, the passenger is transferred into the cab. Otherwise, the unassigned hall call will be recorded again, and re-dispatching might be implemented. After the passenger enters the cab, the array that records the cab call commitments is updated. The passenger waits until the cab arrives at the destination floor and is then transferred to the exit.



(a) The process before a passenger enters a cab



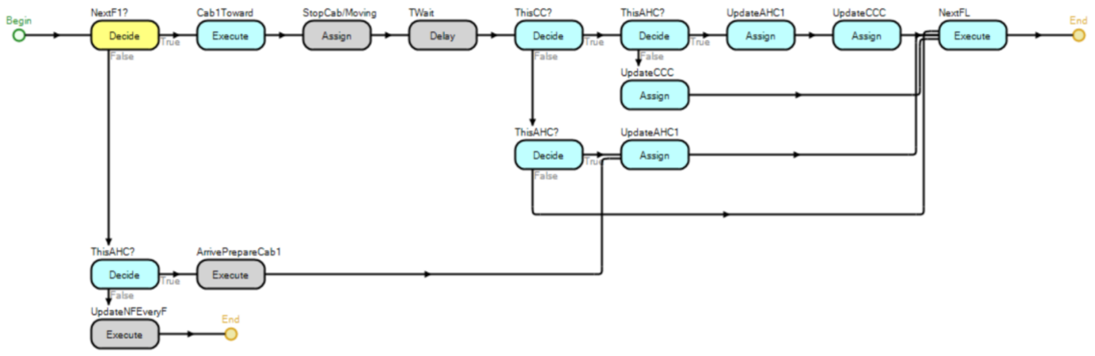
(b) The process after a passenger enters a cab

Figure 8.7 The key process of passenger objects

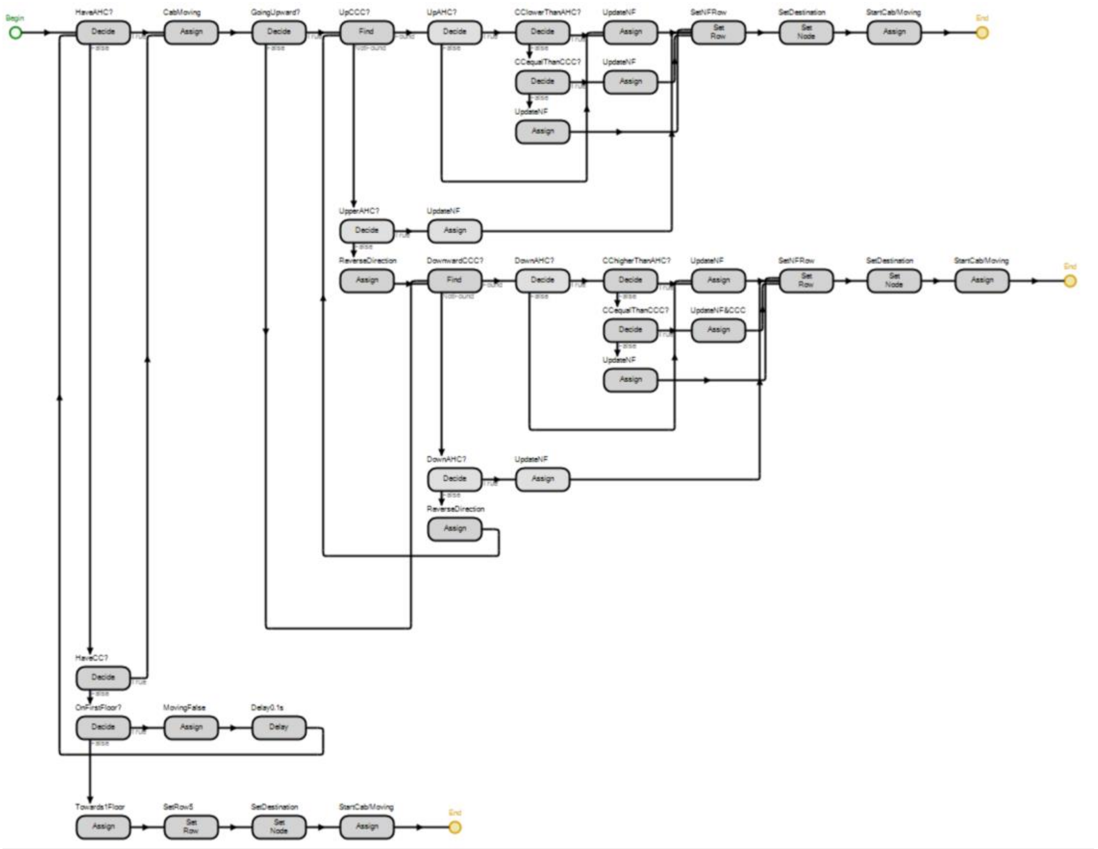
8.6.1.3 Elevator Object Development

The difficulty of modeling elevator objects is to determine the destination of the next task. Because the dispatching operation is dynamic and traffic data keeps changing, assigned hall calls can be reassigned to another cab. Thus, it is necessary to update the next task whenever it arrives at a floor and after it performs a task. In this study, elevator processes are grouped as the floor-related process and the task-related process. The former defines the operation logic when a cab arrives at a floor, while the latter is to find its next task.

Figure 8.8(a) presents the process when a cab arrives at a floor. It first determines if the next task is on this floor or if a hall call from this floor is assigned to the cab when it approaches here. If not, update the next task in case of any change of hall call allocations. If yes, determine the current direction and stop at this floor to drop off or pick up passengers. Find out the task type and update the corresponding array. After the update, determine its next task. Figure 8.8(b) shows the process to determine the next task after a task is performed. Firstly, it checks if the cab has any task. If not, let the cab park on the ground floor. If yes, determine the current direction and if there is any unperformed task in this direction, and these two steps help decide if the change of direction is needed. Then the nearest task is found in that direction as the next task.



(a) The process of a cab arrives at a floor



(b) The process of determining the destination of the next task

Figure 8.8 The key process of elevator objects

8.6.2 *Experiment Design and Input Modelling*

The experiments simulate elevator dispatching in a 10-floor building with two elevator cabs at different traffic intensity levels and aim to analyze the effects of occupancy information on dispatching performance. The facility view of the simulation model is presented in Figure 8.9.

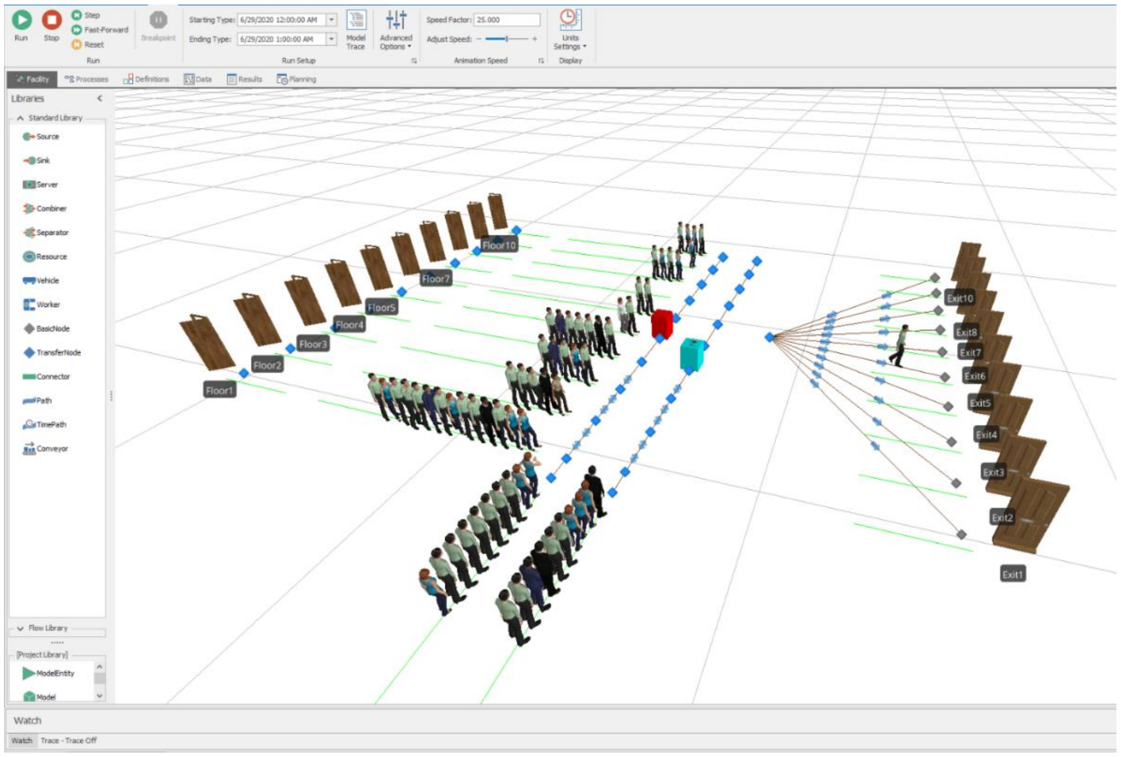

Starting Type: 6/29/2020 12:00:00 AM
Ending Type: 6/29/2020 1:00:00 AM
Speed Factor: 25.000

Figure 8.9 Discrete-event simulation in Simio

There are two groups of experiments. In the first group, variables that record passenger properties include the weight and occupancy area of passengers and their belongings, the passenger destination floor, and the passenger arrival rates. Table 8.5 presents the weight distribution of passengers and other possible objects, and each combination is generated with a predefined probability during the simulation. The occupancy information of these objects can be seen in Table 8.4. The destination floor of a passenger is randomly chosen with the same probability as all other floors. Five travel intensity levels are set, including one scenario for light traffic, normal traffic, and heavy traffic, and two scenarios for intense traffic. Each level has different passenger arrival rates and experiment time, as shown in Table 8.6. Variables regarding elevator cabs are fixed. The occupancy area of the cab ground is set as 4 m^2, and the load capacity is 1000 kg. The time to go to an adjacent floor is 4 seconds. And the time to perform a task is 8 seconds.

Table 8.5 Weight distribution and the probability of passengers and belongings

Combination	Simio Expression		Probability
	Human Weight	Belonging Weight	
Human	Random.Triangular(30, 65, 90)	0	65%
Human + Bicycle	Random.Triangular(30, 65, 90)	Random.Triangular(7.5, 8.5, 10)	2%
Human + Suitcase	Random.Triangular(45, 65, 90)	Random.Triangular(20, 23, 30)	5%
Human + Cardboard Box	Random.Triangular(45, 65, 90)	Random.Triangular(8, 15, 25)	7%
Human + Backpack	Random.Triangular(30, 65, 90)	Random.Triangular(3, 5.5, 8)	20%
Human + Wheelchair	Random.Triangular(50, 65, 90)	Random.Triangular(30, 40, 50)	1%

Table 8.6 Passenger arrival rate settings at different traffic intensity levels

Traffic Intensity	Intensity Level	Simio Expression		Average Number per Hour	Running Time [seconds]
		Ground Floor [minutes]	Upper Floor [minutes]		
I1	Light	Random.Exponential (0.3)	Random.Exponential(1)	740	1800
I2	Normal	Random.Exponential (0.2)	Random.Exponential(2/3)	1110	1200
I3	Heavy	Random.Exponential (0.15)	Random.Exponential(0.5)	1480	900
I4	Intense	Random.Exponential (0.12)	Random.Exponential(0.4)	1850	900
I5	Intense	Random.Exponential (0.1)	Random.Exponential(1/3)	2220	900

To further study the effectiveness of dispatching with occupancy information, the second group of experiments are designed with scenarios where there are more large objects, which is implemented by changing the probability of different scenarios in Table 8.5. The table with the new probability of different object combinations is presented Table 8.7. Also, four different levels of traffic intensities, I_1, I_2, I_3, and I_4, are tested in this group.

Table 8.7 The object generation table with new probabilities

Combination	Simio Expression		Probability
	Human Weight	Belonging Weight	
Human	Random.Triangular(30, 65, 90)	0	20%
Human + Bicycle	Random.Triangular(30, 65, 90)	Random.Triangular(7.5, 8.5, 10)	20%
Human + Suitcase	Random.Triangular(45, 65, 90)	Random.Triangular(20, 23, 30)	20%
Human + Cardboard Box	Random.Triangular(45, 65, 90)	Random.Triangular(8, 15, 25)	20%
Human + Backpack	Random.Triangular(30, 65, 90)	Random.Triangular(3, 5.5, 8)	15%
Human + Wheelchair	Random.Triangular(50, 65, 90)	Random.Triangular(30, 40, 50)	5%

The dispatching optimization problem is solved with the prioritized A* search algorithm. Two dispatching optimization models are developed: one that represents the traditional dispatching methods and only considers the estimated time of arrival as the optimization criterion, and one representing the occupancy-aware dispatching proposed in this paper. The former model has the same algorithm procedure as the latter, and it aims to minimize the average waiting time, while the latter considers both the estimated time of arrival and the cab capacity for dispatching decision evaluation.

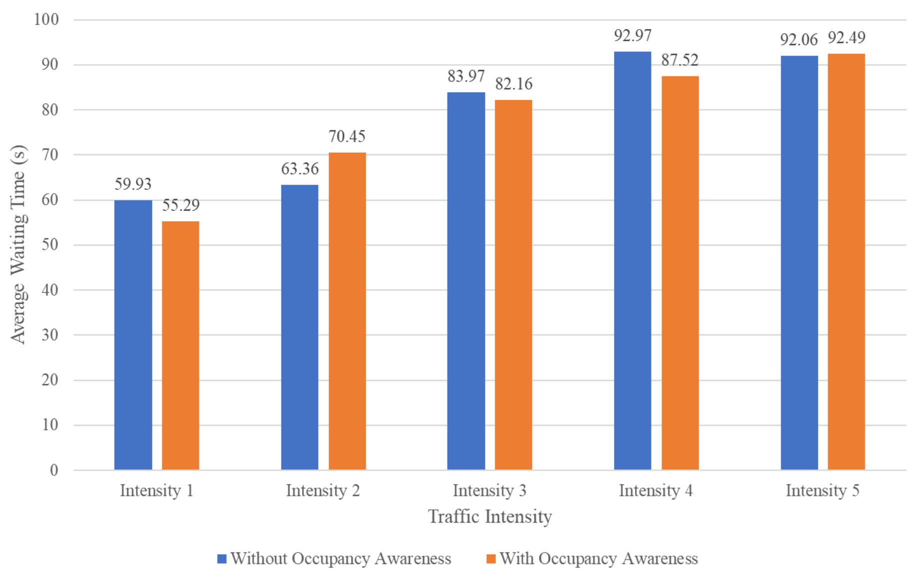
For performance measuring, four indicators are utilized: average waiting time, average travel time, average journey time, and the number of served passengers within the experiment time window. Because the algorithm is designed to minimize the average waiting time, effects of using capacity information on the traffic time are necessary to analyze. Moreover, capacity information conveys the space utilization of a cab, so the quantity of service is another affected perspective.

8.6.3 Experiment Results and Analysis

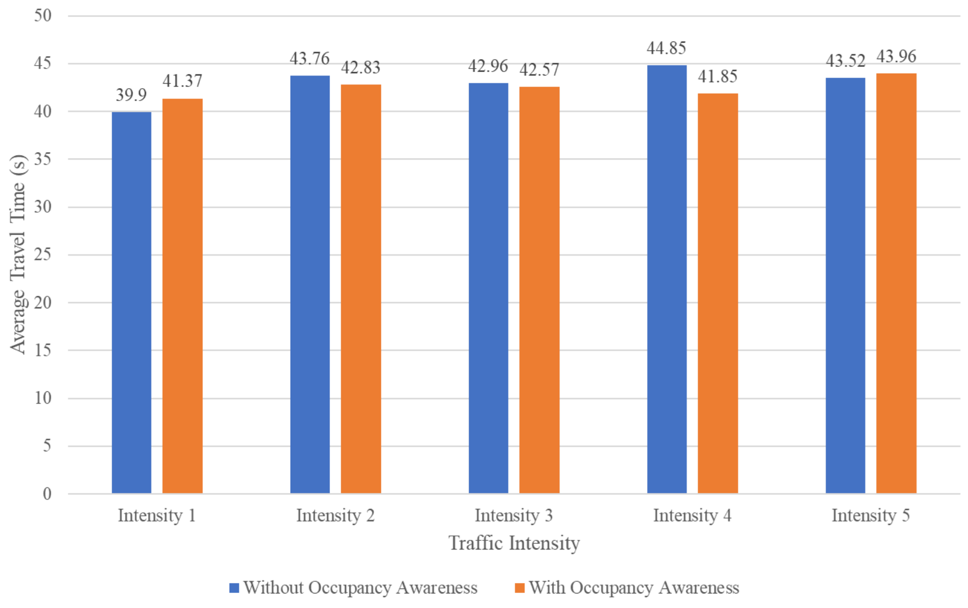
The experiment results of the first group of experiments are recorded in Table 8.8, showing the values of the four performance indicators in two dispatching optimization models at different traffic intensity levels. The comparison charts of the four indicators between the two models are also presented in Figure 8.10.

Table 8.8 Experiment results at different traffic intensity levels

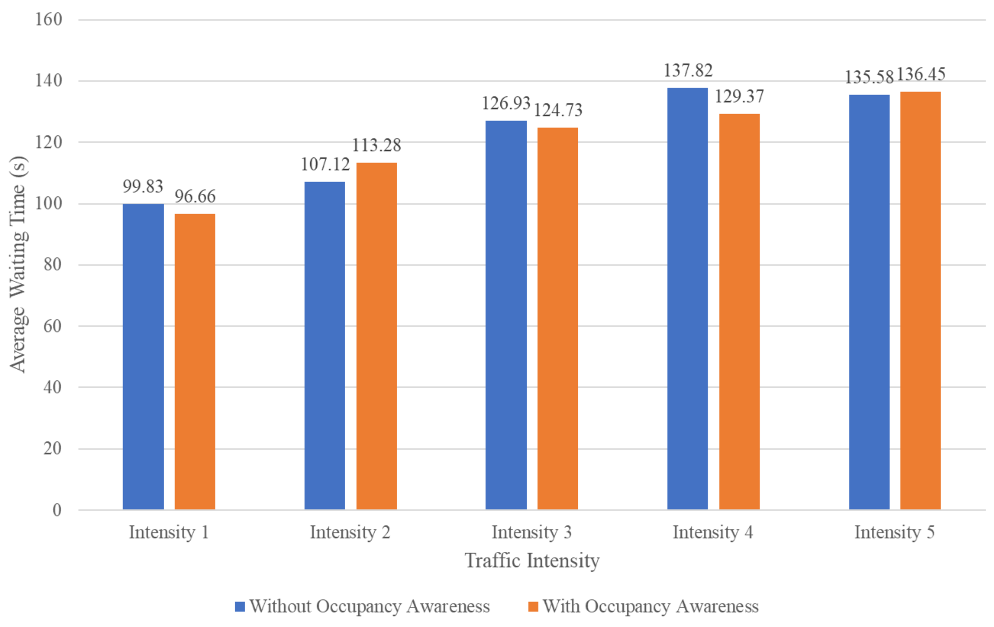
Traffic Intensity	Running Time [Seconds]	Whether Consider Occupancy factor	Average Waiting Time [seconds]	Average Travel Time [seconds]	Average Journey Time [seconds]	Number of Served Passengers
I1	1800	No	59.93	39.90	99.83	358
		Yes	55.29	41.37	96.66	358
I2	1200	No	63.36	43.76	107.12	351
		Yes	70.45	42.83	113.28	364
I3	900	No	83.97	42.96	126.93	308
		Yes	82.16	42.57	124.73	320
I4	900	No	92.97	44.85	137.82	366
		Yes	87.52	41.85	129.37	358
I5	900	No	92.06	43.52	135.58	431
		Yes	92.49	43.96	136.45	450



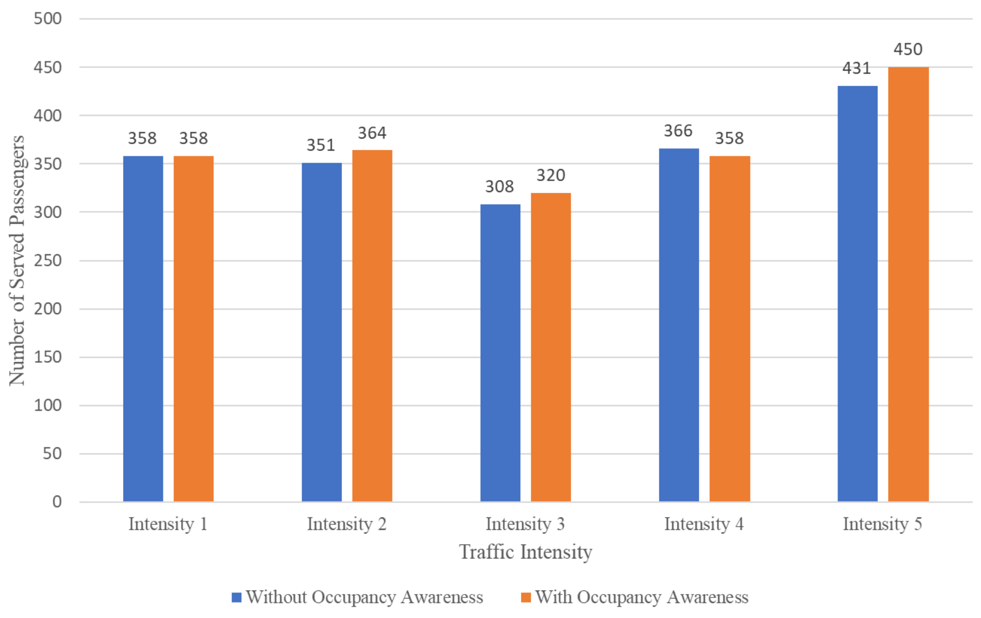
(a) Average waiting time



(b) Average travel time



(c) Average journey time



(d) The number of served passengers

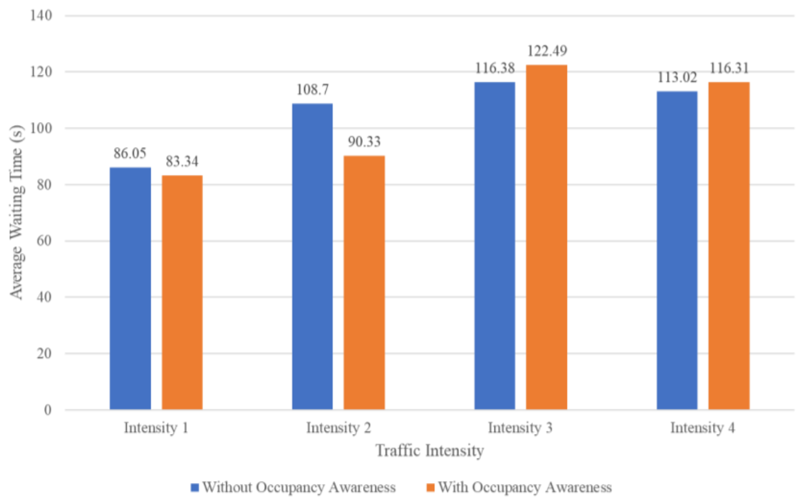
Figure 8.10 Performance comparison on four indicators of the first group

Compared to the traditional dispatching method, the proposed occupancy-aware dispatching fits the occupancy factor into the original objective function as the coefficient of task costs. This change in function structure introduces the trade-off between the time costs and the space utilization of each cab. In Figure 8.10(a), occupancy-aware dispatching has lower average waiting time in light, heavy, and intense traffic. When passenger arrival rates are small, the constraints on capacity can always be satisfied, and pick-up failure will not happen. This characteristic covers up the neglect of capacity in traditional dispatching. In this case, the occupancy factor will influence decision making with cab capacity, and the dispatching optimization model might miss the optimal decision. As passenger arrival rates become larger, capacity is sometimes not enough to accommodate new passengers. Those who fail to enter the cab will spend longer time waiting for the next cab. In this case, the consideration of space utilization helps avoid pick-up failure by skipping the original optimal decision. However, if the traffic intensity becomes too intense, cabs in the system are nearly always fully occupied, and the occupancy factor coefficient on task costs will be similar among different cabs, making the capacity information less effective. Figure 8.10(b) presents the comparison of the average travel time, where occupancy-aware dispatching has better performance in normal, heavy, and intense traffic. The reduction of travel time is because of the reduction in the probability of pick-up failure, which reduces the number of unnecessary stops during a trip. Figure 8.10(c) is the comparison chart of the average journey time, showing the overall effects of the occupancy factor on waiting time and travel time. It suggests occupancy-aware dispatching improves the quality of service in light, heavy, and intense traffic.

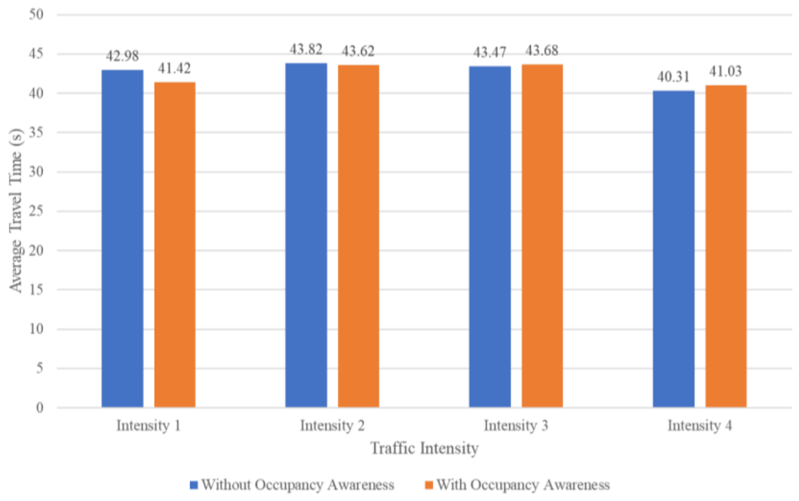
The results of the second group of experiments are shown in Table 8.9. The comparison charts of the four indicators between the two models are also presented in Figure 8.11.

Table 8.9 Results of the second group experiments

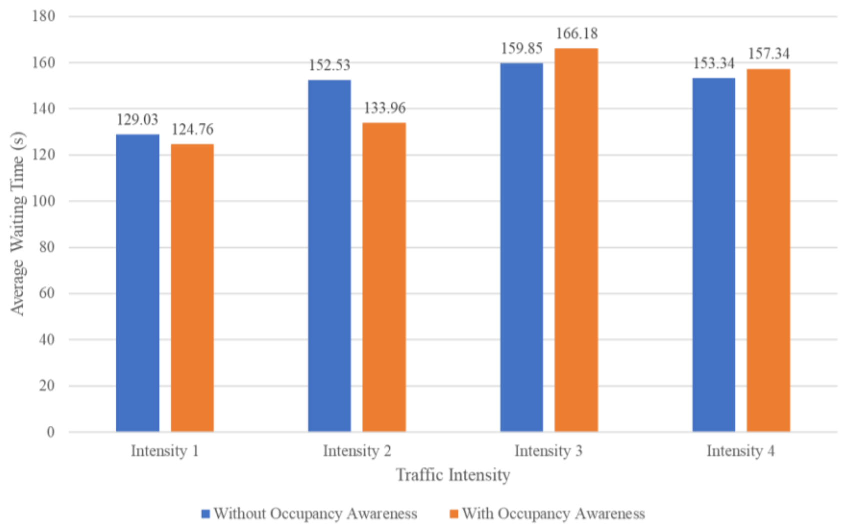
Traffic Intensity	Running Time [Seconds]	Whether Consider Occupancy factor	Average Waiting Time [seconds]	Average Travel Time [seconds]	Average Journey Time [seconds]	Number of Served Passengers
I1	1800	No	86.05	42.98	129.03	357
		Yes	83.34	41.42	124.76	360
I2	1200	No	108.70	43.83	152.53	296
		Yes	90.33	43.63	133.96	302
I3	900	No	116.38	43.47	159.85	254
		Yes	122.49	43.68	166.18	258
I4	900	No	113.02	40.31	153.34	303
		Yes	116.31	41.03	157.34	299



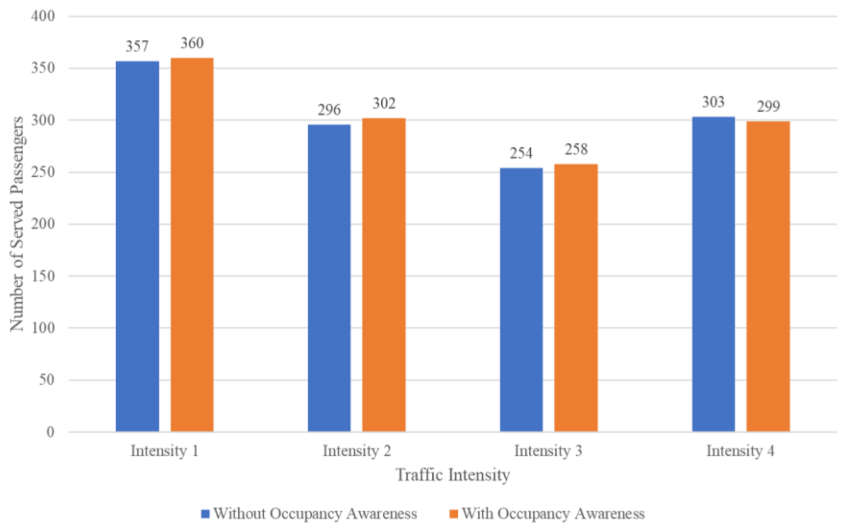
(a) Average waiting time



(b) Average travel time



(c) Average journey time



(d) The number of served passengers

Figure 8.11 Performance comparison on four indicators of the second group

In the second group of experiments, occupancy-aware dispatching has better performance in the quality of service when the traffic intensity is light or normal but has worse performance in heavy and intense traffic intensities. Also, the improvement in the normal traffic intensity is significant. This is because when there are more large objects in the cab, the cab is easily occupied. If the traffic intensity is heavy or intense, the passenger arrival rate is large, and the cab is fully occupied most of the time. In this case, the occupancy factor has the same effects on both cabs. If the traffic intensity is light or normal, the cab is more difficult to get fully occupied, but the capacity constraint is not always satisfied. In such scenarios, occupancy-aware dispatching can effectively avoid pick-up failure and reduce the average waiting time. From the perspective of the quantity of service, occupancy-aware dispatching still has better performance in most traffic intensities: light, normal, and heavy.

To summarize, occupancy-aware dispatching improves both the quality and quantity of service to some extent. It improves the quantity of service if the passenger arrival rate is not too small or too large. It also reduces the average waiting time and the average journey time in most traffic intensities when cabs are sometimes fully occupied in the system.

8.7 Chapter Summary

The validation of the work in this study is through its application to occupancy-aware elevator dispatching optimization, where occupancy awareness refers to cab capacity estimation, usage pattern recognition, and traffic pattern recognition. Mask R-CNN is used to determine the type and number of objects in the environment. Fuzzy logic and case-based reasoning are used to recognize the current elevator usage and the traffic pattern, based on which the dispatching optimization problem can be formulated differently. Estimated capacity is used to represent the penalty cost of a task in the objective function. A prioritized A* search optimization model with occupancy information is developed for real-time dispatching. And the discrete-event simulation model is developed to validate and adjust the use of estimated capacity during dispatching. Experiments show that occupancy-aware dispatching improves the quality and quantity in certain traffic intensities.

CHAPTER 9. CONCLUSIONS

9.1 Contributions

To provide more useful information to the optimization model of a real-time operational system, this thesis proposes to implement CNN-based visual data analysis and intelligent reasoning to extract information from the environment and to deduct applicable knowledge for optimization performance improvement. It enables the operational system to understand the current situation from the image perspective by detection of object existence and reasoning of the underlying information. Also, a closed-loop optimization framework with discrete-event simulation is proposed for optimization model enhancement, in which the functionalities of the simulation are extended to optimization calibration and adaptability adjustment. The validation of this work is done with its application to occupancy-aware elevator dispatching optimization.

In conclusion, the proposed optimization system broadens the information acquisition source and possesses the reasoning ability to make full use of the attainable information. This helps to find solutions of better quality when the decision-making process is conducted in an environment that keeps changing. The use of discrete-event simulation in this system also guarantees the robustness of optimization with situational information by keeping adjusting their interaction.

9.2 Future Work

Future work of this study has three directions. The first direction is to study data augmentation techniques that can generate new information. Because the number of useful data in practice is not always enough, and geometric transformation-based operations only recombine the information, techniques that can generate new information from original data should be studied. The second direction is model verification and validation. Model verification and validation are important steps to guarantee the effectiveness of the work. However, because of the lack of elevator operation data, reasoning-based situational comprehension methods are not verified or validated in this work. The third direction is the study of experimental design in discrete-event simulation. For the research that does not study well-defined problems, the simulation scenarios should reflect the characteristics of the domain problem. Thus, how to design the experiments to represent the problem characteristics should be studied.

REFERENCES

Al-Sharif, L. "Introduction to elevator group control (METE XI)." *Lift Report* 42, no. 2 (2016): 59-68.

Barney, Gina, and Lutfi Al-Sharif. *Elevator traffic handbook: theory and practice*. Routledge, 2015.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations research*, *46*(3), 316-329.

Brachman, R. J., & Levesque, H. J. (2003). *Knowledge representation and Reasoning*. Morgan Kaufmann.

Chan, W. L., Albert TP So, and K. C. Lam. "Dynamic zoning for intelligent supervisory control." *International Journal of Elevator Engineering* 1 (1996): 47-59.

Davis, R., Shrobe, H., & Szolovits, P. (1993). What is a knowledge representation?. *AI magazine*, *14*(1), 17-17.

De Smedt, K. (1988). Knowledge representation techniques in artificial intelligence: An Overview. In *Human-Computer Interaction* (pp. 207-222). Springer, Berlin, Heidelberg.

Dorigo, M., & Di Caro, G. (1999, July). Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)* (Vol. 2, pp. 1470-1477). IEEE.

Dubois, D., & Prade, H. (1996). What are fuzzy rules and how to use them. *Fuzzy sets and systems*, *84*(2), 169-185.

Fernandez, Joaquin R., and Pablo Cortes. "A survey of elevator group control systems for vertical transportation: a look at recent literature." *IEEE Control Systems Magazine* 35, no. 4 (2015): 38-55.

Garey, Michael R., and David S. Johnson. *Computers and intractability*. Vol. 174. San Francisco: freeman, 1979.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

Glover, F., & Laguna, M. (1998). Tabu search. In *Handbook of combinatorial optimization* (pp. 2093-2229). Springer, Boston, MA.

Gomory, R. (1960). *An algorithm for the mixed integer problem*. RAND CORP SANTA MONICA CA.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. Neurocomputing, 187, 27-48.

Hamdi, Muna, and D. J. Mulvaney. "Prioritised A* search in real-time elevator dispatching." *Control Engineering Practice* 15, no. 2 (2007): 219-230.

Hansen, P., & Mladenović, N. (1999). An introduction to variable neighborhood search. In *Meta-heuristics* (pp. 433-458). Springer, Boston, MA.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

He, K., Zhang, X., Ren, S., & Sun, J. (2016, October). Identity mappings in deep residual networks. In *European conference on computer vision* (pp. 630-645). Springer, Cham.

Hertz, A., & Widmer, M. (2003). Guidelines for the use of meta-heuristics in combinatorial optimization. *European Journal of Operational Research, 151*(2), 247-252.

Informs, “What is O.R.?”, //www.informs.org/Explore/What-is-O.R.-Analytics/What-is-O.R. (Accessed November 4, 2020).

JavaTPoint, “Techniques of knowledge representation”, www.javatpoint.com/ai-techniques-of-knowledge-representation (Accessed November 6, 2020).

Kelley, Jr, J. E. (1960). The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, *8*(4), 703-712.

Kelley, Jr, J. E. (1960). The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, *8*(4), 703-712.

Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks* (Vol. 4, pp. 1942-1948). IEEE.

Kim, Chang Bum, Kyoung A. Seong, Hyung Lee-Kwang, Jeong O. Kim, and Yong Bae Lim. "A fuzzy approach to elevator group control system." *IEEE Transactions on systems, man, and cybernetics* 25, no. 6 (1995): 985-990.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671-680.

Kolodner, J. (2014). *Case-based reasoning*. Morgan Kaufmann.

Korf, R. E. (1988). Search: A survey of recent results. In *Exploring artificial intelligence* (pp. 197-237). Morgan Kaufmann.

Korf, Richard E. "Search: A survey of recent results." In *Exploring artificial intelligence*, pp. 197-237. Morgan Kaufmann, 1988.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90.

Lebas, M. J. (1995). Performance measurement and performance management. *International journal of production economics*, *41*(1-3), 23-35.

Levesque, H. J. (1986). Knowledge representation and reasoning. *Annual review of computer science*, *1*(1), 255-287.

Li, Zhonghua, Yunong Zhang, and Hongzhou Tan. "Particle swarm optimization for dynamic sectoring control during peak traffic pattern." In *International Conference on Intelligent Computing*, pp. 650-659. Springer, Berlin, Heidelberg, 2007.

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft coco: Common objects in context." In *European conference on computer vision*, pp. 740-755. Springer, Cham, 2014.

Little, J. D., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations research*, *11*(6), 972-989.

Lu J.J., & Zhang, M. (2013) Heuristic Search. *Encyclopedia of Systems Biology, Springer, New York*, 885-885.

Makridakis, S.G., Wheelwright, S.C. and McGee, V.E. (1983) Forecasting: Methods and Applications. 2nd Edition, Wiley, New York.

OECD. Publishing. (2008). *OECD glossary of statistical terms*. Organisation for Economic Co-operation and Development.

Padberg, M., & Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, *33*(1), 60-100.

Pawlak, Z. (1982). Rough sets. *International journal of computer & information sciences*, *11*(5), 341-356.

Pawlak, Z. (1998). Rough set theory and its applications to data analysis. *Cybernetics & Systems*, *29*(7), 661-688.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

Riesbeck, C. K., & Schank, R. C. (2013). *Inside case-based reasoning*. Psychology Press.

Sabu, M. K., & Raju, G. (2011, March). Rule induction using Rough Set Theory—An application in agriculture. In *2011 International Conference on Computer, Communication and Electrical Technology (ICCCET)* (pp. 45-49). IEEE.

Schrijver, A. (2003). *Combinatorial optimization: polyhedra and efficiency* (Vol. 24). Springer Science & Business Media.

Shin, Hoo-Chang, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M. Summers. "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning." *IEEE transactions on medical imaging* 35, no. 5 (2016): 1285-1298.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data, 6(1), 60.

Siikonen, Marja-Liisa. “Elevator Group Control with Artificial Intelligence.” (1997).

Siikonen, Marja-Liisa. “Elevator Group Control with Artificial Intelligence.” (1997).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556.*

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Wang, F. S., & Chen, L. H. (2013). Heuristic optimization. *Encyclopedia of Systems Biology, Springer, New York*, 885-885.

Watson, I., & Marir, F. (1994). Case-based reasoning: A review. *Knowledge Engineering Review*, *9*(4), 327-354.

Wolsey, L. A. (1998). *Integer programming* (Vol. 52). John Wiley & Sons.

Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, *8*(3), 338-353.

Zadeh, L. A. (1975). Fuzzy logic and approximate reasoning. *Synthese*, *30*(3-4), 407-428.

Zbigniew, S. (2004). An introduction to rough set theory and its applications-A tutorial. Proceedings of ICENCO’2004.